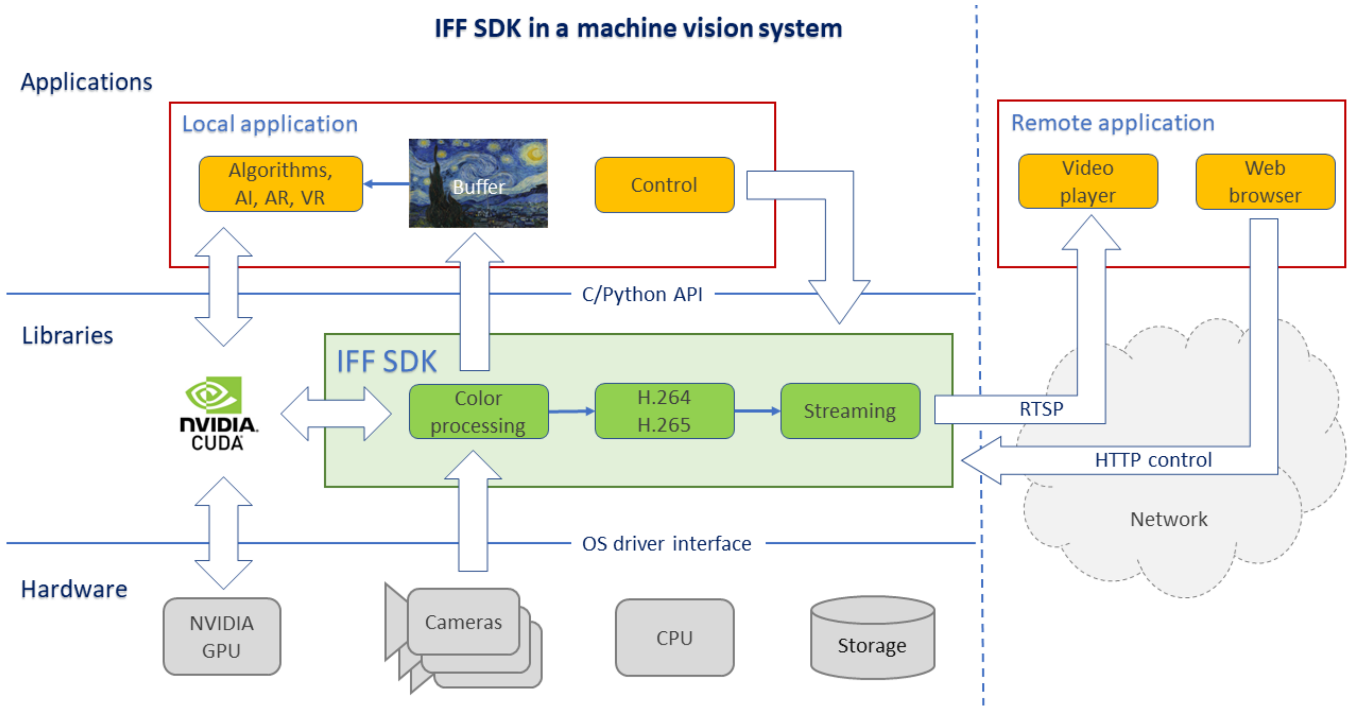
IFF SDK

for high-performance image processing

MRTech **IFF** SDK (**I**mage **F**low **F**ramework SDK) is a powerful cross-platform toolkit that facilitates the development of high-performance machine vision and image processing applications. The core of IFF SDK was initially developed by MRTech in 2016 and has been continuously improved and used in many projects ever since.

The main feature of IFF SDK is delivering images to the customer's application code in the most efficient way. The toolkit helps achieve maximum performance for any configuration of the customer's image processing system.



Basic Functionality

- Textual descriptions of pipeline configurations to build image processing workflows
- Exporting and importing images from the SDK pipeline to the customer application
- Controlling acquisition and processing parameters at runtime
- Seamless integration with the target application using C or Python API as well as REST HTTP interface to control image processing
- Performing accelerated image processing on dedicated and embedded NVIDIA GPUs
- Providing input from multiple machine vision cameras with PCIe, USB3, Ethernet, and MIPI interfaces

Processing modules:
- FFC, white balance, histogram
- Auto-exposure, white balance
- Gamma correction, LUT
- High performance demosaicing
- Spatial denoiser
- Image cropping, resizing
- Color space transformation
- Color correction

Control interfaces:
- JSON configuration
- C/Python API
- HTTP REST API

Compression and decompression:
- H.264, H.265
- JPEG
- JPEG2000

Input capabilities:
- Machine vision cameras
- RTSP video source
- File data input

Output capabilities:
- TIFF/DNG image recording
- RTSP video stream, WebRTC
- Export to user applications

Competitive Advantage

- Production-ready, high-quality code, field-proven in multiple real-life projects
- High-performance image processing with low latency and low overhead
- SDK architecture that makes for easy development and customization of the target application
- Upon request, MRTech provides consulting services and technical support (including implementation assistance)

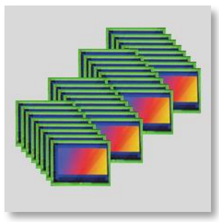
Supported Hardware

- Cameras:
  - Machine vision cameras from XIMEA, Basler, and other vendors
  - MIPI sensors
- Platforms and operating systems:
  - 64-bit Intel x86, Linux and Windows
  - 64-bit ARM, NVIDIA Jetson family, Linux
- Acceleration devices:
  - CUDA processing on NVIDIA GPUs, including Jetson platform
  - Hardware video encoding/decoding on NVIDIA GPUs and Jetsons

Use Cases

High-speed image processing for digital cinema cameras

The developed system includes either one or two high-performance PCIe cameras with **65-megapixel** resolution at up to **70 FPS** and an embedded Intel Core i7 computer.
The MRTech software solution enables the recording of high-framerate, high-resolution clips with **write speed of up to 10 GB/s** and focus on reliability, previewing clips while recording, controlling the camera, and converting recorded images to **CinemaDNG and TIFF** formats.



A multi-camera system for medical applications

The system consists of an array of **32 high-resolution MIPI sensors** placed on a specially designed sensor carrier board and connected to the NVIDIA AGX Xavier module.
The software provides image acquisition with **5 GB/s total bandwidth**, custom image processing pipeline, and an API for remote control.

A large-scale video system for entertainment and 3D scanning

This scalable system is composed of several high-performance Intel/AMD capturing nodes with eight 12-megapixel machine vision cameras per computer.
The software used in the system ensures **64 GB/s bandwidth** for each node.
The entire infrastructure is controlled by a dedicated computer.

Low-latency video solutions for drones and embedded systems

We have a variety of solutions with low latency streaming applications that run on the **NVIDIA Jetson** platform.
The following results can be achieved for the NVIDIA Jetson NX module:

- **1080p (Full HD)** images at 60 FPS with **G2G latency of less than 60 ms**.
- High-dynamic-range **4K UHD** images at 50 FPS with **G2G latency of less than 110 ms**.

For more information and quotations, please contact us at *org@mr-technologies.com*

How to Work with IFF SDK

Follow these steps to tap into all the benefits IFF SDK has to offer:

1. **Select cameras and system hardware**

This step is one of the most important in creating a machine vision system. The right choice of equipment is crucial for project success.

Please contact us if you have any questions. The MRTech team happily shares its extensive hands-on experience, know-hows, and best practices to help our customers successfully implement even the most complex machine vision projects.

2. **Order and get IFF SDK** software package with:
   - Binary libraries for your system configuration
   - C/C++ header files, Python interface module
   - Sample applications with the source code
   - Technical manual in PDF and HTML formats

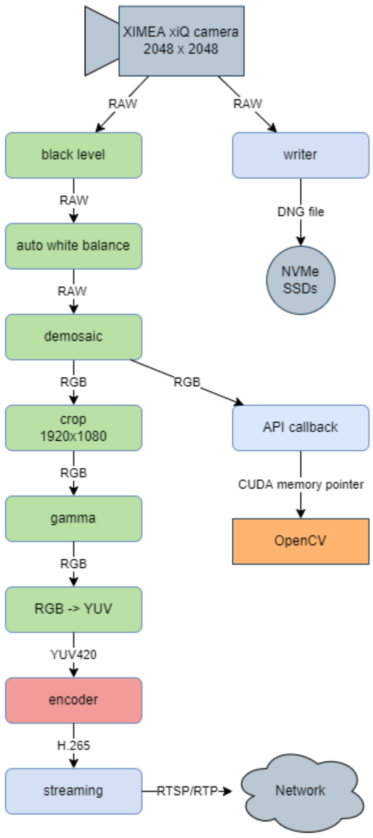
The technical manual or a trial version of IFF SDK can be provided upon request.

3. **Read the technical manual** to explore how to use IFF SDK

The technical manual contains a detailed description of the library components and explains how to use the SDK efficiently.

4. **Try out sample applications** included in the SDK software package and available on GitHub:
   - `farsight` is the first and most general sample application of the SDK.
     It supports various functionalities, including acquisition from a machine vision camera, color pre-processing, auto-exposure and white balance, writing to disk, H.265 encoding, and RTSP streaming.
   - `imagebroker` is another sample application that shows how to export images to the user code across IFF SDK library boundaries. Additionally, it provides an example code to render an image on the screen using OpenCV.

5. **Design the required image processing pipeline** using IFF SDK components and describe the pipeline in the JSON format.
   The figure on the right represents an example pipeline.



6. **Build your own application**

Below is an example of a basic C application that illustrates the user-friendly, low-code approach implemented in IFF SDK interface:

```
iff_initialize(base_cfg);
void* handle = iff_create_chain(chain_cfg, error_handler);
iff_chain_exec_cmd(handle, "streaming/on", "");
```

For more information, follow the links below or contact us via email:

MRTech IFF SDK	https://mr-te.ch/sdk
GitHub account	https://github.com/mr-technologies
MRTech support	support@mr-technologies.com