inivation-docs

Release 2026-11-14

iniVation AG

2026-11-14

HARDWARE

1 Current Products 2

2 Discontinued Products 54

3 Advanced Usage 56

4 Introduction 91

5 DV 93

6 DV-Processing 157

7 Other Software 158

8 Advanced Usage 159

9 Frequently Asked Questions 184

10 Support 192

11 Other Resources 193

Please also consult our main web page, inivation.com[1].

[1] https://inivation.com/

CHAPTER

ONE

CURRENT PRODUCTS

1.1 DVXplorer

Buy Device[2]

Download PDF[3]



Date: 2026-11-14

- Mass production DVS sensor in 90 nm BSI CIS technology
- VGA (640x480) resolution event output with up to 110 dB dynamic range, sub 1ms latency, 200 µs temporal resolution and up to 165 million events per second throughput
- 6-axis IMU, up to 3.2 kHz gyroscope sampling rate, up to 1.6 kHz accelerometer sampling rate
- Supports multi-camera time synchronization via daisy chain connection and external event injection
- Consumes less than 140 mA at 5 V power supply
- Anodized aluminium case with CS lens mount, 4-side mounting options.
- Screw-locked USB port and fully isolated multi-camera sync ports

1.1.1 Specifications

[2] https://inivation.com/buy/
[3] https://docs.inivation.com/_static/hardware_guides/dvxplorer.pdf

Event Output

Description	Value
Spatial Resolution	640 x 480
Temporal Resolution [1]	200 µs
Typical Latency [2]	<1 ms
Max Throughput	165 MEPS
Dynamic Range	~ 90 dB (3-100k lux with 99.9% of pixels respond to 27.5% contrast), ~ 110 dB (0.3-100k lux with 50% of pixels respond to 80% contrast)
Contrast Sensitivity	13% (with 50% of pixels respond), 27.5% (with 99.9% of pixels respond)

[1] The temporal resolution is characterized by the timestamp unit. In fact, a timestamp unit of 1 us offers minimum gain over a timestamp unit of 200 us. For more explanation, please refer to our *white paper*.

[2] The temporal latency is given as a nominal number and can be improved with strong lighting or optimised biases.

IMU

6-axis (Gyro + Accelerometer), up to 3.2 kHz gyroscope sampling rate, up to 1.6 kHz accelerometer sampling rate. The IMU is synchronized with the event and frame output. Read more in the *IMU section*.

Camera Synchronization / Trigger Input

Supports *multi-camera time synchronization* via daisy chain connection and external event injection.

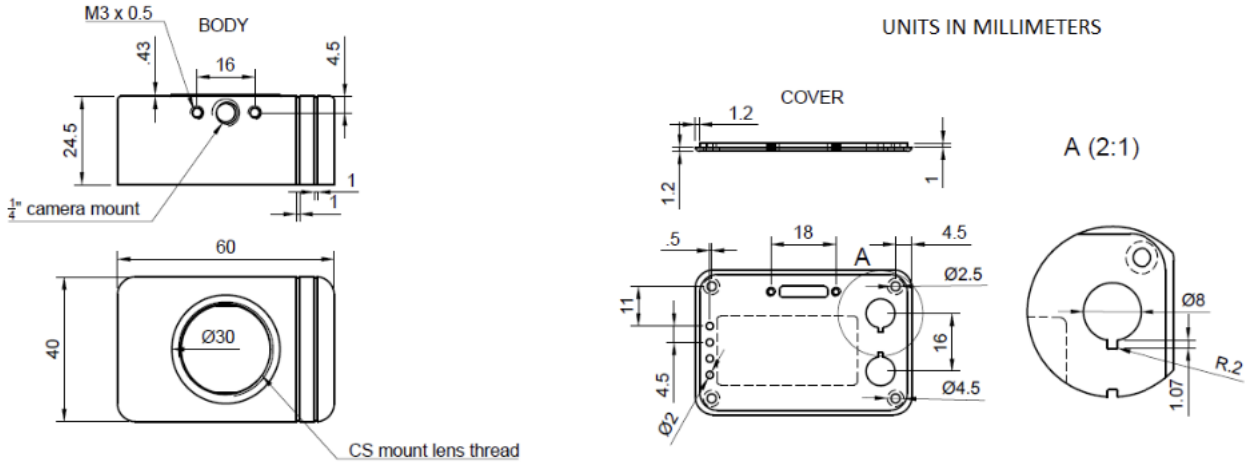
Other Attributes

Description	Value
Dimensions [mm]	W 40 x H 60 x D 25
Weight	100 g without lens
Lens Mount	CS-mount
Case Material	Anodized aluminium
Mounting Options	4-side Whitworth 1/4”-20 female and M3 mounting points
Connectors	USB 3.0 micro B port with locking screws, fully isolated sync input and output connectors
Multi-cam sync	Yes
Power Consumption	<140 mA @ 5 VDC (USB)
Sensor Technology	90 nm BSI CIS
Pixel Pitch [µm]	9 µm
Sensor Supply Voltage	1.2V, 1.8 V and 2.8 V
Certifications	CE

Specifications not guaranteed. All specifications subject to change without notice.

1.1.2 Physical Dimensions

The DVXplorer camera is housed in an anodized aluminum case. The case dimensions are depicted below.



1.1.3 Connectors

DVXplorer has three connectors on the back. One USB 3.0 Micro B connector for data and power, and two sync connectors for multi-camera synchronization or external trigger input.



USB 3.0 Connector

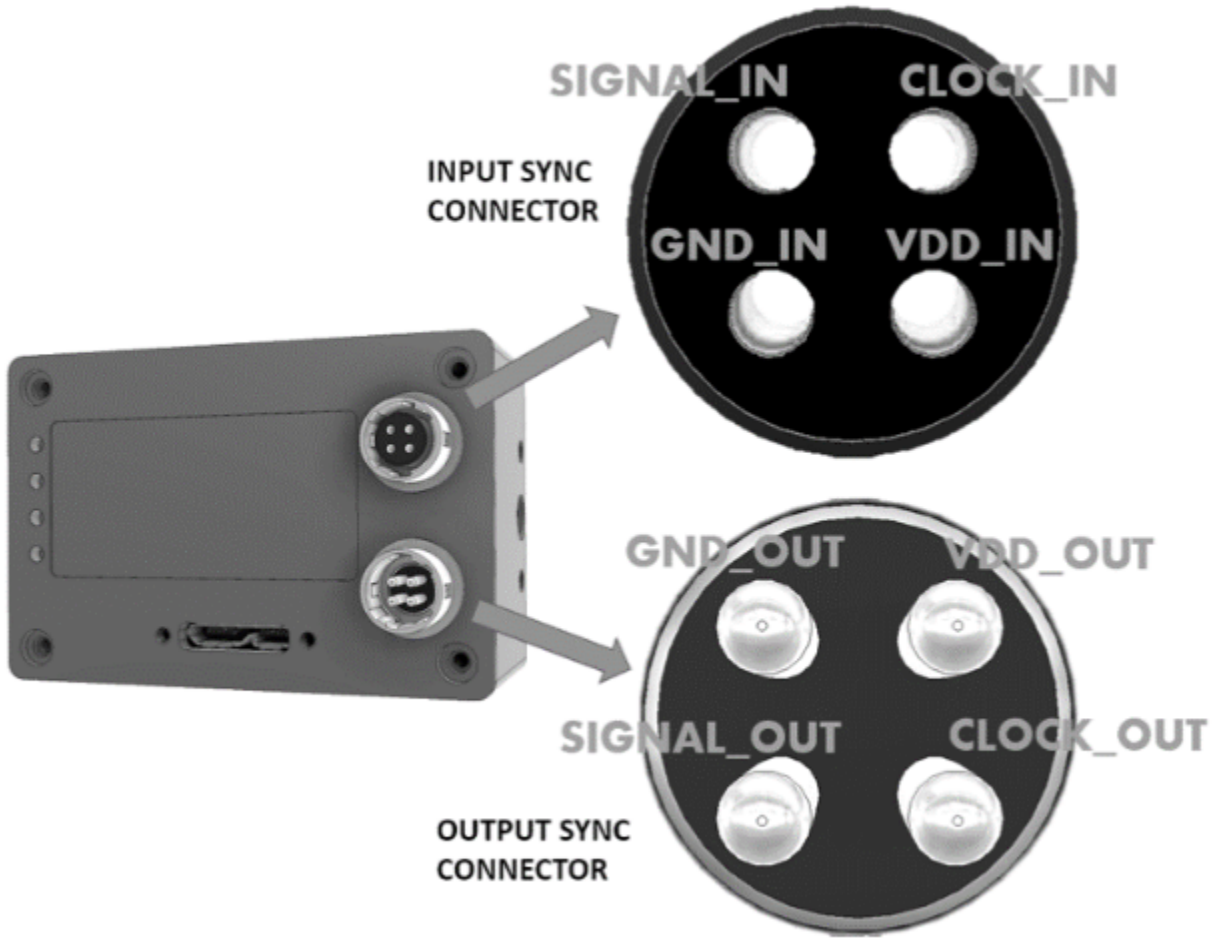
The USB 3.0 Micro B connector is used for data and power. Any USB 3.0 or USB 2.0 cable with Micro B type connector can be used. However, USB 3.0 speeds are only supported when using a USB 3.0 cable. Usage of cables with appropriate locking screws are recommended for a more secure and robust connection.

Sync Connectors

The synchronization connectors are HiRose HR10A-7R-4P (male, SYNC OUTPUT) and HR10A-7R-4S (female, SYNC INPUT) connectors. Cables should use the matching connectors HR10A-7P-4S (female) and HR10A-7P-4P (male).

Please note that to keep full electrical isolation between different cameras, the cable should not be shielded, or if it is, the shield should not connect one end of the cable to the other.

Input signals can be 3.3V or 5V, depending on the VDD_IN supplied externally, output signals are 5V, as is VDD_OUT. If you chain cameras together for synchronization, the clock and VDD will be 5V, for example.



Synchronization connectors pinout is shown in the image above. Please note that all the pins in the SYNC IN ports are isolated from the SYNC OUT ports.

1.1.4 Optics

The camera lens mount is designed to accommodate CS-mount lenses. Other lenses can be accommodated by using adapters. The standard lens shipped with the camera is a C-mount lens and ships with an adapter. The chip requires a lens designed for 1/2.5" imagers.

The field of view (FOV) depends on the focal length **L** of the lens and the size **W** of the pixel array. It is computed from geometrical optics, not accounting for any lens distortion. The angular field of view (**AFOV**) is given by:

$$AFOV = 2 \arctan(\frac{W}{2L})$$

The linear FOV (**LFOV**) at a distance **D** from the lens is given by:

$$LFOV = \frac{D * W}{L}$$

The pixel array has a resolution of 640 x 480 and measures:

- Width: 640 pixels x 9.0 µm/pixel = 5.76 mm
- Height: 480 pixels x 9.0 µm/pixel = 4.32 mm
- Diagonal: 7.20 mm

Computed Field of View

The following table shows the horizontal and vertical field of view in degrees and its size at various distances for different common focal lengths.

Lens Focal Length [mm]	Horizontal Angular FoV [deg]	Vertical Angular FoV [deg]	Diagonal Angular FoV [deg]	Horizontal Linear FoV at 10 cm distance [cm]	Horizontal Linear FoV at 30 cm distance [cm]	Horizontal Linear FoV at 1 m distance [cm]	Horizontal Linear FoV at 2 m distance [cm]
2.10	107.80	91.61	119.49	27.43	82.29	274.29	548.57
2.80	91.61	75.30	104.25	20.57	61.71	205.71	411.43
3.00	87.66	71.51	100.39	19.20	57.60	192.00	384.00
3.60	77.32	61.93	90.00	16.00	48.00	160.00	320.00
4.50	65.24	51.28	77.32	12.80	38.40	128.00	256.00
6.00	51.28	39.60	61.93	9.60	28.80	96.00	192.00
9.00	35.49	26.99	43.60	6.40	19.20	64.00	128.00
12.00	26.99	20.41	33.40	4.80	14.40	48.00	96.00
16.00	20.41	15.38	25.36	3.60	10.80	36.00	72.00

1.1.5 Additional Information

Software

DVXplorer is compatible with all our software. You can use it in:

- *DV software*
- *dv-processing*

Serial Number

The serial number of the device can be found on the case, usually a five-digit number printed on a label located on the back of the camera case.

Package Contents

DVXplorer ships with the following items:

- DVXplorer camera
- USB 3.0, 1m with locking screws
- Varifocal C mount lens (Datasheet[4])
- CS to C mount lens adapter
- Tripod

Safety Information

To prevent damage to property or injury to yourself or to others, read this safety information in its entirety before using this product.

- This product is intended to be used in a laboratory and for industrial applications under controlled conditions.

[4] https://docs.inivation.com/_static/lenses/vari-focal-lens-incl.pdf

- We strongly recommend that you only use high quality USB cables, like the ones provided by iniVation. Using low quality USB cables could cause damages to the device.
- Keep the product dry. Do not handle the product with wet hands. Do not handle the plug with wet hands. Do not operate the camera near water. This could cause damage to the device. The camera is not water-safe.
- Handling: Handle your product with care. It is made of metal, glass, and plastic and has sensitive electronic components inside. The product can be damaged if dropped, burned, punctured, or crushed, or if it comes in contact with liquid. If you suspect damage to the product, please contact iniVation.
- Repairing: Do not open the product and do not attempt to repair the product yourself. Disassembling the product may damage it and will void your warranty. If your product is damaged or malfunctions, please contact iniVation.
- Do not disassemble or modify this product.
- Do not touch internal parts that become exposed as the result of a fall or other accident.
- Keep this product out of reach of children. Should a child swallow any part of this product, seek immediate medical attention.
- Use travel converters or adapters designed to convert from one voltage to another or with DC-to-AC inverters.
- Explosive and other atmospheric conditions. Connecting or using the product in any area with a potentially explosive atmosphere, such as areas where the air contains high levels of flammable chemicals, vapors, or particles (such as grain, dust, or metal powders), may be hazardous. Exposing the product to environments which have high concentrations of industrial chemicals, including near evaporating liquified gasses such as helium, may damage or impair the product's functionality.
- Turn this product off when its use is prohibited.
- Do not leave the product where it will be exposed to elevated temperatures for an extended period such as in an enclosed automobile or in direct sunlight. This can lead to malfunction.

Correct Disposal





This product and its electronic accessories should not be disposed of with other household waste. If you are unable to dispose of this item safely please return it to iniVation AG.

CE Certification


iniVation declares that this device is CE certified. This product is not ATEX approved.

1.2 DVXplorer Lite

Buy Device[5]

Download PDF[6]



Date: 2026-11-14

- Mass production DVS sensor in 90 nm BSI CIS technology
- QVGA (320x240) resolution event output with up to 110 dB dynamic range, sub 1ms latency, 200 µs temporal resolution and up to 100 million events per second throughput
- 6-axis IMU, up to 3.2 kHz gyroscope sampling rate, up to 1.6 kHz accelerometer sampling rate
- Supports multi-camera time synchronization via daisy chain connection and external event injection
- Plastic case with CS lens mount, 4-side mounting options.
- Screw-locked USB port and fully isolated multi-camera sync ports

[5] https://inivation.com/buy/
[6] https://docs.inivation.com/_static/hardware_guides/dvxplorer-lite.pdf

1.2.1 Specifications

Event Output

Description	Value
Spatial Resolution	320 x 240
Temporal Resolution [1]	200 µs
Typical Latency [2]	<1 ms
Max Throughput	100 MEPS
Dynamic Range	~ 90 dB (3-100k lux with 99.9% of pixels respond to 27.5% contrast), ~ 110 dB (0.3-100k lux with 50% of pixels respond to 80% contrast)
Contrast Sensitivity	13% (with 50% of pixels respond), 27.5% (with 99.9% of pixels respond)

[1] The temporal resolution is characterized by the timestamp unit. In fact, a timestamp unit of 1 us offers minimum gain over a timestamp unit of 200 us. For more explanation, please refer to our *white paper*.

[2] The temporal latency is given as a nominal number and can be improved with strong lighting or optimised biases.

IMU

6-axis (Gyro + Accelerometer), up to 3.2 kHz gyroscope sampling rate, up to 1.6 kHz accelerometer sampling rate. The IMU is synchronized with the event and frame output. Read more in the *IMU section*.

Camera Synchronization / Trigger Input

Supports *multi-camera time synchronization* via daisy chain connection and external event injection.

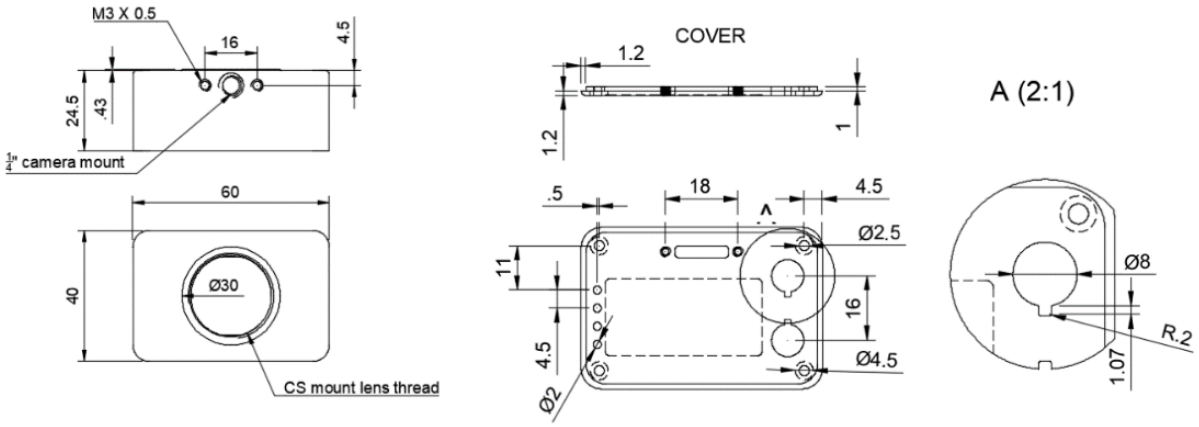
Other Attributes

Description	Value
Dimensions [mm]	W 40 x H 60 x D 25
Weight	100 g without lens
Lens Mount	CS-mount
Case Material	POM (plastic)
Mounting Options	4-side Whitworth 1/4”-20 female and M3 mounting points
Connectors	USB 3.0 micro B port with locking screws, fully isolated sync input and output connectors
Multi-cam sync	Yes
Power Consumption	<140 mA @ 5 VDC (USB)
Sensor Technology	90 nm BSI CIS
Pixel Pitch [µm]	18 µm
Sensor Supply Voltage	1.2V, 1.8 V and 2.8 V
Certifications	CE

Specifications not guaranteed. All specifications subject to change without notice.

1.2.2 Physical Dimensions

The DVXplorer Lite camera is housed in a POM plastic case. The case dimensions are depicted below.



1.2.3 Connectors

DVXplorer Lite has three connectors on the back. One USB 3.0 Micro B connector for data and power, and two sync connectors for multi-camera synchronization or external trigger input.



USB 3.0 Connector

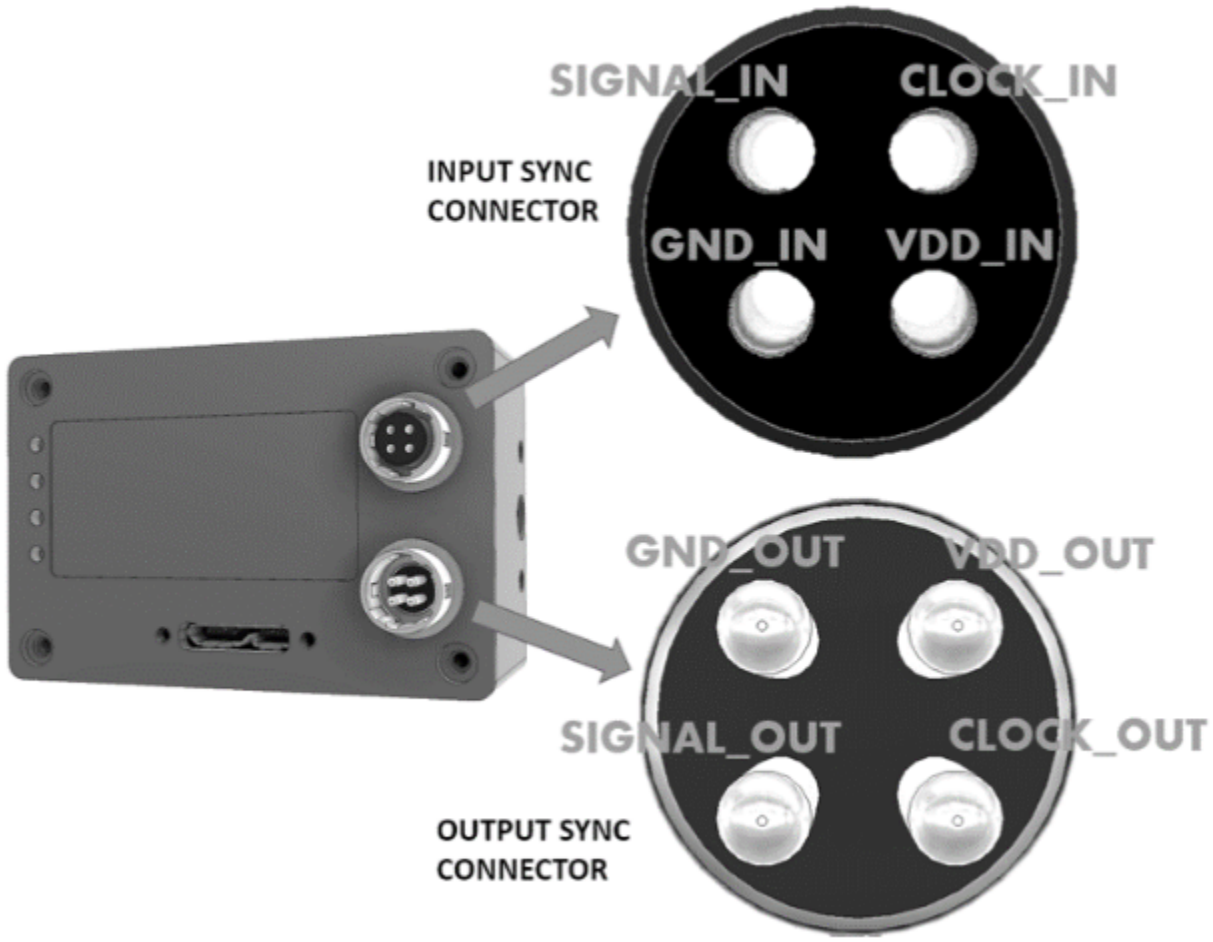
The USB 3.0 Micro B connector is used for data and power. Any USB 3.0 or USB 2.0 cable with Micro B type connector can be used. However, USB 3.0 speeds are only supported when using a USB 3.0 cable. Usage of cables with appropriate locking screws are recommended for a more secure and robust connection.

Sync Connectors

The synchronization connectors are HiRose HR10A-7R-4P (male, SYNC OUTPUT) and HR10A-7R-4S (female, SYNC INPUT) connectors. Cables should use the matching connectors HR10A-7P-4S (female) and HR10A-7P-4P (male).

Please note that to keep full electrical isolation between different cameras, the cable should not be shielded, or if it is, the shield should not connect one end of the cable to the other.

Input signals can be 3.3V or 5V, depending on the VDD_IN supplied externally, output signals are 5V, as is VDD_OUT. If you chain cameras together for synchronization, the clock and VDD will be 5V, for example.



Synchronization connectors pinout is shown in the image above. Please note that all the pins in the SYNC IN ports are isolated from the SYNC OUT ports.

1.2.4 Optics

The camera lens mount is designed to accommodate CS-mount lenses. Other lenses can be accommodated by using adapters. The standard lens shipped with the camera is a C-mount lens and ships with an adapter. The chip requires a lens designed for 1/2.5" imagers.

The field of view (FOV) depends on the focal length **L** of the lens and the size **W** of the pixel array. It is computed from geometrical optics, not accounting for any lens distortion. The angular field of view (**AFOV**) is given by:

$$AFOV = 2 \arctan(\frac{W}{2L})$$

The linear FOV (**LFOV**) at a distance **D** from the lens is given by:

$$LFOV = \frac{D * W}{L}$$

The pixel array has a resolution of 320 x 240 and measures:

- Width: 320 pixels x 18 µm/pixel = 5.76 mm
- Height: 240 pixels x 18 µm/pixel = 4.32 mm
- Diagonal: 7.20 mm

Computed Field of View

The following table shows the horizontal and vertical field of view in degrees and its size at various distances for different common focal lengths.

Lens Focal Length [mm]	Horizontal Angular FoV [deg]	Vertical Angular FoV [deg]	Diagonal Angular FoV [deg]	Horizontal Linear FoV at 10 cm distance [cm]	Horizontal Linear FoV at 30 cm distance [cm]	Horizontal Linear FoV at 1 m distance [cm]	Horizontal Linear FoV at 2 m distance [cm]
2.10	107.80	91.61	119.49	27.43	82.29	274.29	548.57
2.80	91.61	75.30	104.25	20.57	61.71	205.71	411.43
3.00	87.66	71.51	100.39	19.20	57.60	192.00	384.00
3.60	77.32	61.93	90.00	16.00	48.00	160.00	320.00
4.50	65.24	51.28	77.32	12.80	38.40	128.00	256.00
6.00	51.28	39.60	61.93	9.60	28.80	96.00	192.00
9.00	35.49	26.99	43.60	6.40	19.20	64.00	128.00
12.00	26.99	20.41	33.40	4.80	14.40	48.00	96.00
16.00	20.41	15.38	25.36	3.60	10.80	36.00	72.00

1.2.5 Additional Information

Software

DVXplorer Lite is compatible with all our software. You can use it in:

- *DV software*
- *dv-processing*

Serial Number

The serial number of the device can be found on the case, usually a five-digit number printed on a label located on the back of the camera case.

Package Contents

DVXplorer Lite ships with the following items:

- DVXplorer Lite camera
- USB 3.0, 1m with locking screws
- Varifocal C mount lens (Datasheet[7])
- CS to C mount lens adapter
- Tripod

Safety Information

To prevent damage to property or injury to yourself or to others, read this safety information in its entirety before using this product.

- This product is intended to be used in a laboratory and for industrial applications under controlled conditions.

[7] https://docs.inivation.com/_static/lenses/vari-focal-lens-incl.pdf

- We strongly recommend that you only use high quality USB cables, like the ones provided by iniVation. Using low quality USB cables could cause damages to the device.
- Keep the product dry. Do not handle the product with wet hands. Do not handle the plug with wet hands. Do not operate the camera near water. This could cause damage to the device. The camera is not water-safe.
- Handling: Handle your product with care. It is made of metal, glass, and plastic and has sensitive electronic components inside. The product can be damaged if dropped, burned, punctured, or crushed, or if it comes in contact with liquid. If you suspect damage to the product, please contact iniVation.
- Repairing: Do not open the product and do not attempt to repair the product yourself. Disassembling the product may damage it and will void your warranty. If your product is damaged or malfunctions, please contact iniVation.
- Do not disassemble or modify this product.
- Do not touch internal parts that become exposed as the result of a fall or other accident.
- Keep this product out of reach of children. Should a child swallow any part of this product, seek immediate medical attention.
- Use travel converters or adapters designed to convert from one voltage to another or with DC-to-AC inverters.
- Explosive and other atmospheric conditions. Connecting or using the product in any area with a potentially explosive atmosphere, such as areas where the air contains high levels of flammable chemicals, vapors, or particles (such as grain, dust, or metal powders), may be hazardous. Exposing the product to environments which have high concentrations of industrial chemicals, including near evaporating liquified gasses such as helium, may damage or impair the product's functionality.
- Turn this product off when its use is prohibited.
- Do not leave the product where it will be exposed to elevated temperatures for an extended period such as in an enclosed automobile or in direct sunlight. This can lead to malfunction.

Correct Disposal





This product and its electronic accessories should not be disposed of with other household waste. If you are unable to dispose of this item safely please return it to iniVation AG.

CE Certification


iniVation declares that this device is CE certified. This product is not ATEX approved.

1.3 DVXplorer Micro

Buy Device[8]

Download PDF[9]



Date: 2026-11-14

- Mass production DVS sensor in 90 nm BSI CIS technology
- VGA (640x480) resolution event output with up to 110 dB dynamic range, sub 1ms latency, 200 µs temporal resolution and up to 450 million events per second throughput
- 6-axis IMU, up to 3.2 kHz gyroscope sampling rate, up to 1.6 kHz accelerometer sampling rate
- Consumes less than 140 mA at 5 V USB power supply
- Plastic case with S lens mount, 2-side mounting options and screw-locked USB port

1.3.1 Specifications

[8] https://inivation.com/buy/
[9] https://docs.inivation.com/_static/hardware_guides/dvxplorer-micro.pdf

Event Output

Description	Value
Spatial Resolution	640 x 480
Temporal Resolution [1]	200 µs
Typical Latency [2]	<1 ms
Max Throughput	450 MEPS
Dynamic Range	~ 90 dB (3-100k lux with 99.9% of pixels respond to 27.5% contrast), ~ 110 dB (0.3-100k lux with 50% of pixels respond to 80% contrast)
Contrast Sensitivity	13% (with 50% of pixels respond), 27.5% (with 99.9% of pixels respond)

[1] The temporal resolution is characterized by the timestamp unit. In fact, a timestamp unit of 1 us offers minimum gain over a timestamp unit of 200 us. For more explanation, please refer to our *white paper*.

[2] The temporal latency is given as a nominal number and can be improved with strong lighting or optimised biases.

IMU

6-axis (Gyro + Accelerometer), up to 3.2 kHz gyroscope sampling rate, up to 1.6 kHz accelerometer sampling rate. The IMU is synchronized with the event and frame output. Read more in the *IMU section*.

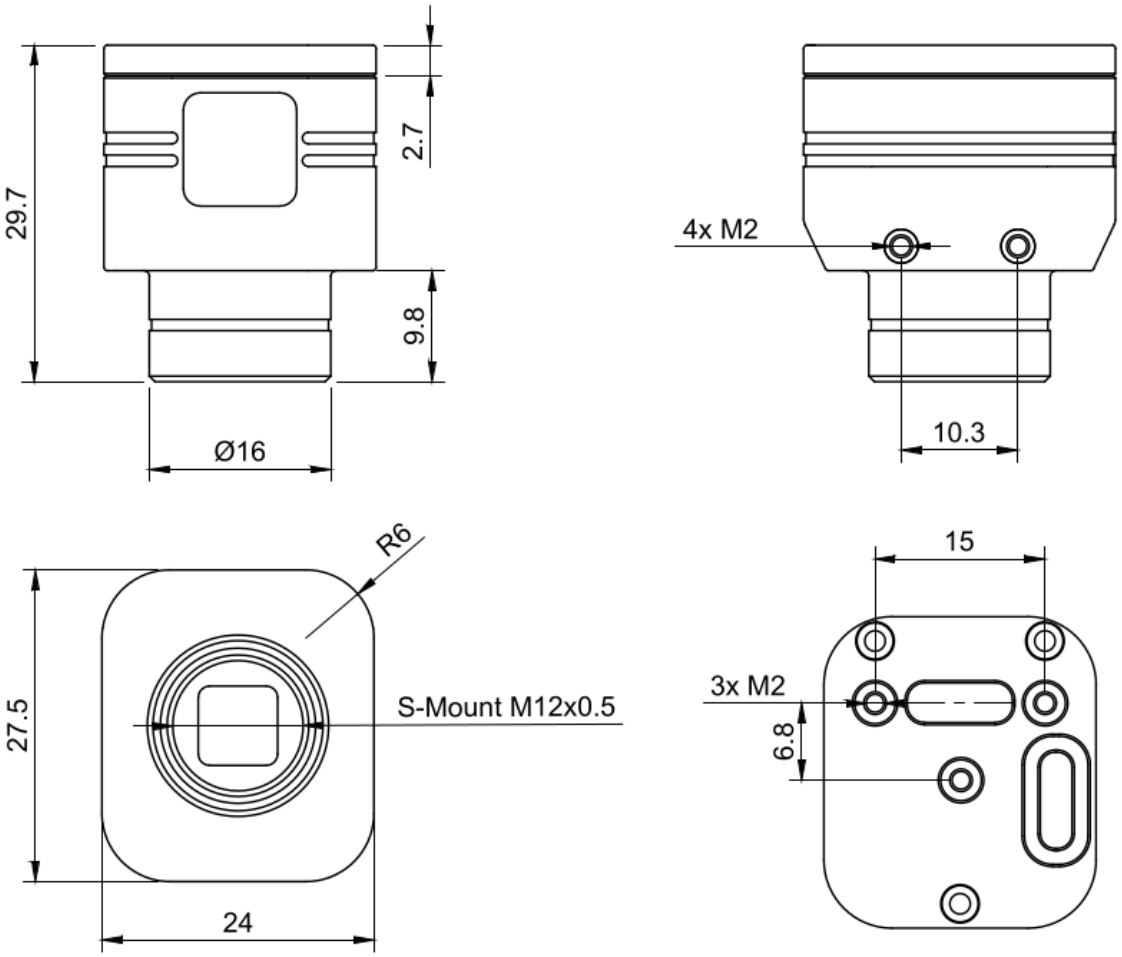
Other Attributes

Description	Value
Dimensions [mm]	W 24 x H 27.5 x D 29.7
Weight	16g without lens
Lens Mount	S-mount
Case Material	POM (plastic)
Mounting Options	2-side M2 mounting points
Connectors	USB 3.1 Type-C
Multi-cam sync	None
Power Consumption	<140 mA @ 5 VDC (USB)
Sensor Technology	90 nm BSI CIS
Pixel Pitch [µm]	9 µm
Sensor Supply Voltage	1.2V, 1.8 V and 2.8 V

Specifications not guaranteed. All specifications subject to change without notice.

1.3.2 Physical Dimensions

The DVXplorer Micro camera is housed in a POM plastic case. The case dimensions are depicted below.



1.3.3 Connectors

DVXplorer Micro has one connector on the back. A USB 3.1 Type-C connector for data and power.



USB C Connector

The USB C connector is used for data and power. Any USB 3.0 or USB 2.0 cable with Type-C connector can be used. However, USB 3.0 speeds are only supported when using a USB 3.0 cable. Usage of cables with appropriate locking screws are recommended for a more secure and robust connection.

1.3.4 Optics

The camera lens mount is designed to accommodate S-mount lenses. Other lenses can be accommodated by using adapters. The chip requires a lens designed for 1/2.5" imagers.

The field of view (FOV) depends on the focal length **L** of the lens and the size **W** of the pixel array. It is computed from geometrical optics, not accounting for any lens distortion. The angular field of view (**AFOV**) is given by:

$$AFOV = 2 \arctan(\frac{W}{2L})$$

The linear FOV (**LFOV**) at a distance **D** from the lens is given by:

$$LFOV = \frac{D * W}{L}$$

The pixel array has a resolution of 640 x 480 and measures:

- Width: 640 pixels x 9.0 µm/pixel = 5.76 mm
- Height: 480 pixels x 9.0 µm/pixel = 4.32 mm
- Diagonal: 7.20 mm

Computed Field of View

The following table shows the horizontal and vertical field of view in degrees and its size at various distances for different common focal lengths.

Lens Focal Length [mm]	Horizontal Angular FoV [deg]	Vertical Angular FoV [deg]	Diagonal Angular FoV [deg]	Horizontal Linear FoV at 10 cm distance [cm]	Horizontal Linear FoV at 30 cm distance [cm]	Horizontal Linear FoV at 1 m distance [cm]	Horizontal Linear FoV at 2 m distance [cm]
2.10	107.80	91.61	119.49	27.43	82.29	274.29	548.57
2.80	91.61	75.30	104.25	20.57	61.71	205.71	411.43
3.00	87.66	71.51	100.39	19.20	57.60	192.00	384.00
3.60	77.32	61.93	90.00	16.00	48.00	160.00	320.00
4.50	65.24	51.28	77.32	12.80	38.40	128.00	256.00
6.00	51.28	39.60	61.93	9.60	28.80	96.00	192.00
9.00	35.49	26.99	43.60	6.40	19.20	64.00	128.00
12.00	26.99	20.41	33.40	4.80	14.40	48.00	96.00
16.00	20.41	15.38	25.36	3.60	10.80	36.00	72.00

1.3.5 Additional Information

Software

DVXplorer Micro is compatible with all our software. You can use it in:

- *DV software*
- *dv-processing*

Serial Number

The serial number of the device can be found on the case, usually with **"DXU"** followed by a four-digit number printed on a label located at the bottom of the camera case.

Package Contents

DVXplorer Micro ships with the following items:

- DVXplorer Micro camera
- USB 3.1, A male to C male cable, 1m with locking screws
- 3 different S-Mount (M12) lenses
  - 3.6 mm (Datasheet[10])
  - 6 mm (Datasheet[11])
  - 16 mm (Datasheet[12])
- S-Mount lockring

Safety Information

To prevent damage to property or injury to yourself or to others, read this safety information in its entirety before using this product.

- This product is intended to be used in a laboratory and for industrial applications under controlled conditions.
- We strongly recommend that you only use high quality USB cables, like the ones provided by iniVation. Using low quality USB cables could cause damages to the device.
- Keep the product dry. Do not handle the product with wet hands. Do not handle the plug with wet hands. Do not operate the camera near water. This could cause damage to the device. The camera is not water-safe.
- Handling: Handle your product with care. It is made of metal, glass, and plastic and has sensitive electronic components inside. The product can be damaged if dropped, burned, punctured, or crushed, or if it comes in contact with liquid. If you suspect damage to the product, please contact iniVation.
- Repairing: Do not open the product and do not attempt to repair the product yourself. Disassembling the product may damage it and will void your warranty. If your product is damaged or malfunctions, please contact iniVation.
- Do not disassemble or modify this product.
- Do not touch internal parts that become exposed as the result of a fall or other accident.
- Keep this product out of reach of children. Should a child swallow any part of this product, seek immediate medical attention.
- Use travel converters or adapters designed to convert from one voltage to another or with DC-to-AC inverters.
- Explosive and other atmospheric conditions. Connecting or using the product in any area with a potentially explosive atmosphere, such as areas where the air contains high levels of flammable chemicals, vapors, or particles (such as grain, dust, or metal powders), may be hazardous. Exposing the product to environments which have high concentrations of industrial chemicals, including near evaporating liquified gasses such as helium, may damage or impair the product's functionality.
- Turn this product off when its use is prohibited.
- Do not leave the product where it will be exposed to elevated temperatures for an extended period such as in an enclosed automobile or in direct sunlight. This can lead to malfunction.

[10] https://docs.inivation.com/_static/lenses/micro-lens-3-6-incl.pdf
[11] https://docs.inivation.com/_static/lenses/micro-lens-6-0-incl.pdf
[12] https://docs.inivation.com/_static/lenses/micro-lens-16-incl.pdf

Correct Disposal





This product and its electronic accessories should not be disposed of with other household waste. If you are unable to dispose of this item safely please return it to iniVation AG.

1.3.6 Certifications

DVXplorer Micro is currently not certified for any purpose.

1.4 DAVIS346

Buy Device[13]

Download PDF[14]



Date: 2026-11-14

- Prototype DAVIS sensor in 180 nm CIS technology
- Concurrent QVGA+ (346 x 260) resolution event and frame output from single sensor
- Event output with up to 120 dB dynamic range, sub 1 µs latency, 1 µs temporal resolution and up to 12 million events per second throughput
- 6-axis IMU, up to 8 kHz sampling rate
- Supports multi-camera time synchronization via daisy chain connection and external event injection
- Consumes less than 180 mA at 5 V power supply
- Anodized aluminium case with CS lens mount, 4-side mounting options

[13] https://inivation.com/buy/
[14] https://docs.inivation.com/_static/hardware_guides/davis346.pdf

- Screw-locked USB port and fully isolated multi-camera sync ports

1.4.1 Specifications

Event Output

Description	Value
Spatial Resolution	346 x 260
Temporal Resolution [1]	1 µs
Typical Latency [2]	<1 ms
Max Throughput	12 MEPS
Dynamic Range	~ 120 dB (0.1-100k lux with 50% of pixels respond to 80% contrast)
Contrast Sensitivity	14.3% (on) 22.5% (off) (with 50% of pixels respond)

[1] The temporal resolution is characterized by the timestamp unit. In fact, a timestamp unit of 1 us offers minimum gain over a timestamp unit of 200 us. For more explanation, please refer to our *white paper*.

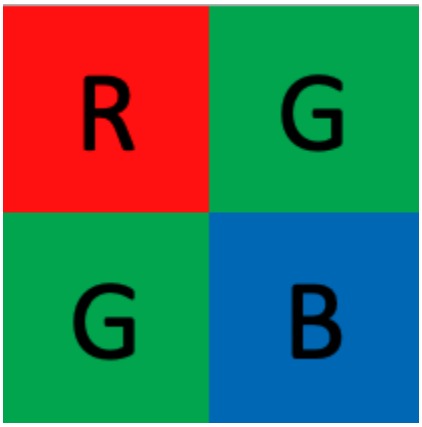
[2] The temporal latency is given as a nominal number and can be improved with strong lighting or optimised biases.

Frame Output

Description	Value
Spatial Resolution	346 x 260
Frame Rate	40 FPS
Dynamic Range	55 dB
FPN	4.2%
Dark Signal	18000 e-/s
Readout Noise	55 e-

Color Event and Frame Output

We provide both a mono (grayscale) and a color version of DAVIS 346. The mono version is a monochrome sensor and the color version is a color sensor that uses a Bayer color filter (RGGB as seen in the image below) for both events and frames. By default, the camera uses demosaicing for reconstructing RGB frames. In order to change to raw color output, please refer to *DV software* or *dv-processing*. Note that the color frames are not calibrated, and thus do not faithfully reproduce the real observed color.



IMU

6-axis (Gyro + Accelerometer), up to 8 kHz sampling rate. The IMU is synchronized with the event and frame output. Read more in the *IMU section*.

Camera Synchronization / Trigger Input

Supports *multi-camera time synchronization* via daisy chain connection and external event injection.

Other Attributes

Description	Value
Dimensions [mm]	W 40 x H 60 x D 25
Weight	100 g without lens
Lens Mount	CS-mount
Case Material	Anodized aluminium
Mounting Options	4-side Whitworth 1/4”-20 female and M3 mounting points
Connectors	USB 3.0 micro B port with locking screws, fully isolated sync input and output connectors
Multi-cam sync	Yes
Power Consumption	<180 mA @ 5 VDC (USB)
Sensor Technology	0.18 µm 1P6M MIM CIS
Pixel Pitch [µm]	18.5 µm
Sensor Supply Voltage	1.8 V and 3.3 V
Certifications	CE

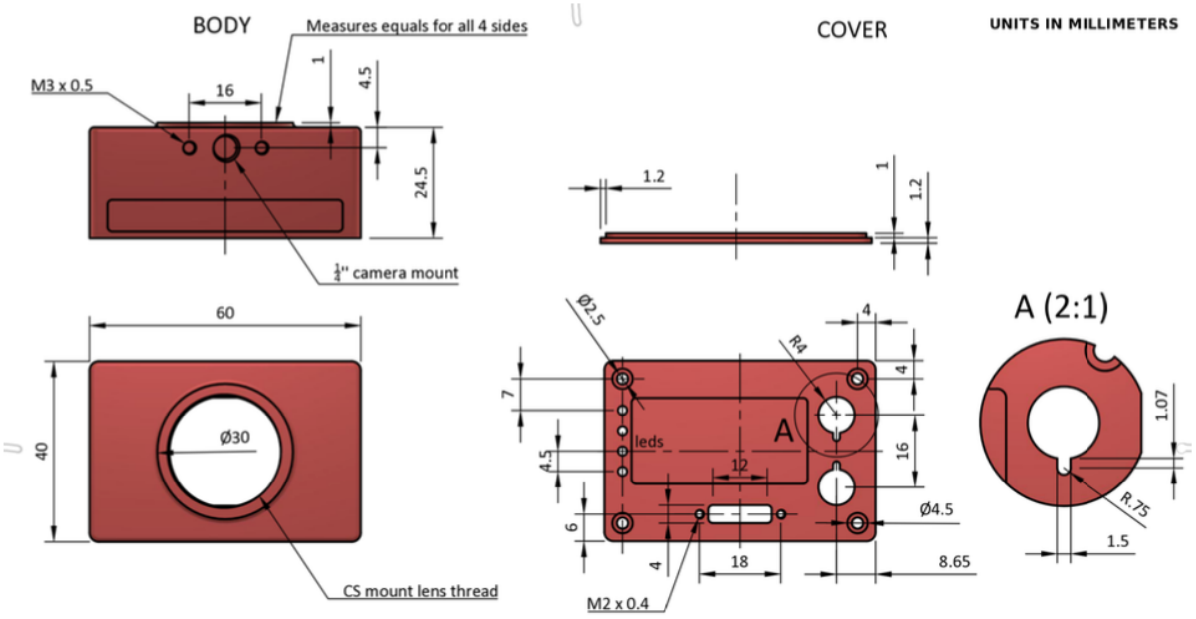
Specifications not guaranteed. All specifications subject to change without notice.

1.4.2 Sensor Limitations

- In APS GlobalShutter mode, bursts of DVS events can be caused by the capture of an APS frame.
- Due to bandwidth limitations, the DVS event output tends to follow a scanning pattern when under high load.
- The frame output has below average performance in terms of image quality compared to conventional image sensors.
- Color frames are not calibrated, and thus do not faithfully reproduce the real observed color.
- The event output can be destabilized if a strong light source impacts a sensitive spot outside the photosensitive pixel array.

1.4.3 Physical Dimensions

The DAVIS camera is housed in an anodized aluminum case. The case dimensions are depicted below.



1.4.4 Connectors

DAVIS has three connectors on the back. One USB 3.0 Micro B connector for data and power, and two sync connectors for multi-camera synchronization or external trigger input.



USB 3.0 Connector

The USB 3.0 Micro B connector is used for data and power. Any USB 3.0 or USB 2.0 cable with Micro B type connector can be used. However, USB 3.0 speeds are only supported when using a USB 3.0 cable. Usage of cables with appropriate locking screws are recommended for a more secure and robust connection.

Sync Connectors

The synchronization connectors are HiRose HR10A-7R-4P (male, SYNC OUTPUT) and HR10A-7R-4S (female, SYNC INPUT) connectors. Cables should use the matching connectors HR10A-7P-4S (female) and HR10A-7P-4P (male).

Please note that to keep full electrical isolation between different cameras, the cable should not be shielded, or if it is, the shield should not connect one end of the cable to the other.

Input signals can be 3.3V or 5V, depending on the VDD_IN supplied externally, output signals are 5V, as is VDD_OUT. If you chain cameras together for synchronization, the clock and VDD will be 5V, for example.



Synchronization connectors pinout is shown in the image above. Please note that all the pins in the SYNC IN ports are isolated from the SYNC OUT ports.

1.4.5 Optics

The camera lens mount is designed to accommodate CS-mount lenses. Other lenses can be accommodated by using adapters. The standard lens shipped with the camera is a C-mount lens and ships with an adapter. The chip requires a lens designed for 1/2" imagers.

The field of view (FOV) depends on the focal length **L** of the lens and the size **W** of the pixel array. It is computed from geometrical optics, not accounting for any lens distortion. The angular field of view (**AFOV**) is given by:

$$AFOV = 2 \arctan(\frac{W}{2L})$$

The linear FOV (**LFOV**) at a distance **D** from the lens is given by:

$$LFOV = \frac{D * W}{L}$$

The pixel array has a resolution of 346 x 260 and measures:

- Width: 346 pixels x 18.5 µm/pixel = 6.4 mm
- Height: 260 pixels x 18.5 µm/pixel = 4.81 mm
- Diagonal: 8 mm

Field of View Computations

The following table shows the horizontal and vertical field of view in degrees and its size at various distances for different common focal lengths.

Lens Focal Length [mm]	Horizontal Angular FoV [deg]	Vertical Angular FoV [deg]	Diagonal Angular FoV [deg]	Horizontal Linear FoV at 10 cm distance [cm]	Horizontal Linear FoV at 30 cm distance [cm]	Horizontal Linear FoV at 1 m distance [cm]	Horizontal Linear FoV at 2 m distance [cm]
2.10	113.46	97.75	124.64	30.48	91.44	304.81	609.62
2.80	97.64	81.32	110.06	22.86	68.58	228.61	457.21
3.00	93.70	77.44	106.31	21.34	64.01	213.37	426.73
3.60	83.28	67.49	96.07	17.78	53.34	177.81	355.61
4.50	70.84	56.24	83.32	14.22	42.67	142.24	284.49
6.00	56.15	43.69	67.43	10.67	32.01	106.68	213.37
9.00	39.15	29.92	47.96	7.11	21.34	71.12	142.24
12.00	29.87	22.67	36.90	5.33	16.00	53.34	106.68
16.00	22.62	17.10	28.10	4.00	12.00	40.01	80.01

1.4.6 Additional Information

Software

DAVIS 346 is compatible with all our software. You can use it in:

- *DV software*
- *dv-processing*

Serial Number

The serial number of the device can be found on the case, usually a four-digit number printed on a black label located at the top of the camera case.

Package Contents

DAVIS 346 ships with the following items:

- DAVIS 346 camera
- USB 3.0, 1m with locking screws
- Varifocal C mount lens (Datasheet[15])
- CS to C mount lens adapter
- Tripod

Safety Information

To prevent damage to property or injury to yourself or to others, read this safety information in its entirety before using this product.

- This product is intended to be used in a laboratory and for industrial applications under controlled conditions.

[15] https://docs.inivation.com/_static/lenses/vari-focal-lens-incl.pdf

- We strongly recommend that you only use high quality USB cables, like the ones provided by iniVation. Using low quality USB cables could cause damages to the device.
- Keep the product dry. Do not handle the product with wet hands. Do not handle the plug with wet hands. Do not operate the camera near water. This could cause damage to the device. The camera is not water-safe.
- Handling: Handle your product with care. It is made of metal, glass, and plastic and has sensitive electronic components inside. The product can be damaged if dropped, burned, punctured, or crushed, or if it comes in contact with liquid. If you suspect damage to the product, please contact iniVation.
- Repairing: Do not open the product and do not attempt to repair the product yourself. Disassembling the product may damage it and will void your warranty. If your product is damaged or malfunctions, please contact iniVation.
- Do not disassemble or modify this product.
- Do not touch internal parts that become exposed as the result of a fall or other accident.
- Keep this product out of reach of children. Should a child swallow any part of this product, seek immediate medical attention.
- Use travel converters or adapters designed to convert from one voltage to another or with DC-to-AC inverters.
- Explosive and other atmospheric conditions. Connecting or using the product in any area with a potentially explosive atmosphere, such as areas where the air contains high levels of flammable chemicals, vapors, or particles (such as grain, dust, or metal powders), may be hazardous. Exposing the product to environments which have high concentrations of industrial chemicals, including near evaporating liquified gasses such as helium, may damage or impair the product's functionality.
- Turn this product off when its use is prohibited.
- Do not leave the product where it will be exposed to elevated temperatures for an extended period such as in an enclosed automobile or in direct sunlight. This can lead to malfunction.

Correct Disposal





This product and its electronic accessories should not be disposed of with other household waste. If you are unable to dispose of this item safely please return it to iniVation AG.

CE Certification


iniVation declares that this device is CE certified. This product is not ATEX approved.

1.5 DAVIS346 AER

Buy Device[16]

Download PDF[17]



Date: 2026-11-14

- Prototype DAVIS sensor in 180 nm CIS technology
- Concurrent QVGA+ (346 x 260) resolution event and frame output from single sensor via USB
- Event-only output via AER connector
- Event output with up to 120 dB dynamic range, sub 1 µs latency, 1 µs temporal resolution and up to 12 million events per second throughput
- 6-axis IMU, up to 8 kHz sampling rate
- Consumes less than 180 mA at 5 V power supply
- Anodized aluminium case with CS lens mount, 4-side mounting options
- Screw-locked USB port and fully isolated multi-camera sync ports

1.5.1 Specifications

[16] https://inivation.com/buy/
[17] https://docs.inivation.com/_static/hardware_guides/davis346-aer.pdf

Event Output

Description	Value
Spatial Resolution	346 x 260
Temporal Resolution [1]	1 µs
Typical Latency [2]	<1 ms
Max Throughput	12 MEPS
Dynamic Range	~ 120 dB (0.1-100k lux with 50% of pixels respond to 80% contrast)
Contrast Sensitivity	14.3% (on) 22.5% (off) (with 50% of pixels respond)

[1] The temporal resolution is characterized by the timestamp unit. In fact, a timestamp unit of 1 us offers minimum gain over a timestamp unit of 200 us. For more explanation, please refer to our *white paper*.

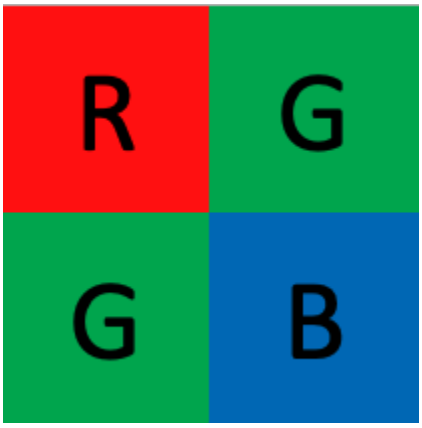
[2] The temporal latency is given as a nominal number and can be improved with strong lighting or optimised biases.

Frame Output

Description	Value
Spatial Resolution	346 x 260
Frame Rate	40 FPS
Dynamic Range	55 dB
FPN	4.2%
Dark Signal	18000 e-/s
Readout Noise	55 e-

Color Event and Frame Output

We provide both a mono (grayscale) and a color version of DAVIS 346. The mono version is a monochrome sensor and the color version is a color sensor that uses a Bayer color filter (RGGB as seen in the image below) for both events and frames. By default, the camera uses demosaicing for reconstructing RGB frames. In order to change to raw color output, please refer to *DV software* or *dv-processing*. Note that the color frames are not calibrated, and thus do not faithfully reproduce the real observed color.



IMU

6-axis (Gyro + Accelerometer), up to 8 kHz sampling rate. The IMU is synchronized with the event and frame output. Read more in the *IMU section*.

Camera Synchronization / Trigger Input

Multi-camera synchronization and external triggers are not supported. The AER output does not have a concept of time or timestamping, so there is also no concept of synchronizing multiple cameras directly. You can do this on your own FPGA as needed.

Other Attributes

Description	Value
Dimensions [mm]	W 40 x H 60 x D 25
Weight	120 g without lens
Lens Mount	CS-mount
Case Material	Anodized aluminium
Mounting Options	4-side Whitworth 1/4”-20 female and M3 mounting points
Connectors	USB 3.0 micro B port with locking screw, AER connector with 2.54 mm pins
Multi-cam sync	No
Power Consumption	<180 mA @ 5 VDC (USB)
Sensor Technology	0.18 µm 1P6M MIM CIS
Pixel Pitch [µm]	18.5 µm
Sensor Supply Voltage	1.8 V and 3.3 V
Certifications	None

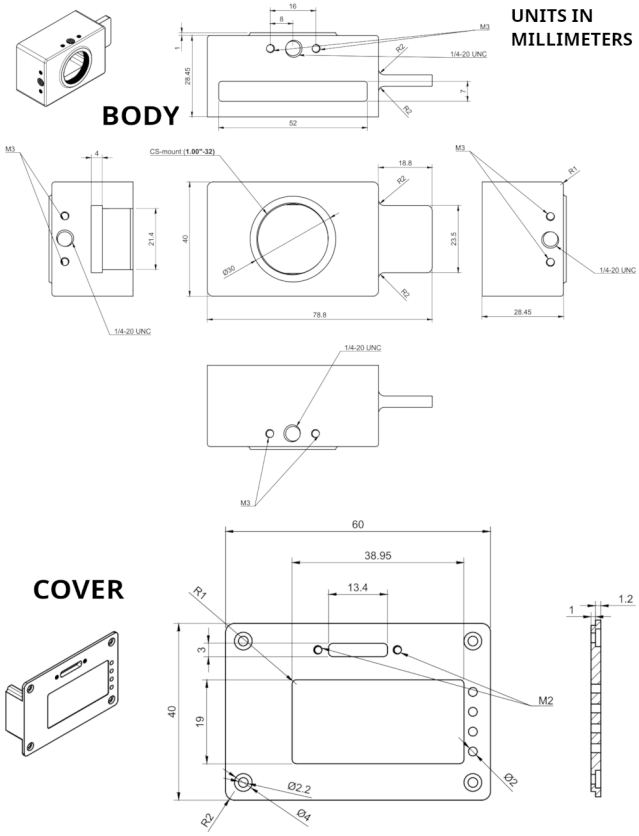
Specifications not guaranteed. All specifications subject to change without notice.

1.5.2 Sensor Limitations

- In APS GlobalShutter mode, bursts of DVS events can be caused by the capture of an APS frame.
- Due to bandwidth limitations, the DVS event output tends to follow a scanning pattern when under high load.
- The frame output has below average performance in terms of image quality compared to conventional image sensors.
- Color frames are not calibrated, and thus do not faithfully reproduce the real observed color.
- The event output can be destabilized if a strong light source impacts a sensitive spot outside the photosensitive pixel array.
- The AER connector can only transmit events, not frames or IMU data.
- No multi-camera timestamp synchronization is present, nor external triggers.

1.5.3 Physical Dimensions

The DAVIS camera is housed in an anodized aluminum case. The case dimensions are depicted below.



1.5.4 Connectors

DAVIS 346 AER has one USB 3.0 connector for data and power on its back, as well as an AER connector to the side. Please note no timestamp synchronization and/or trigger connectors are provided on this model!

USB 3.0 Connector

The USB 3.0 Micro B connector is used for data and power. Any USB 3.0 or USB 2.0 cable with Micro B type connector can be used. However, USB 3.0 speeds are only supported when using a USB 3.0 cable. Usage of cables with appropriate locking screws are recommended for a more secure and robust connection.

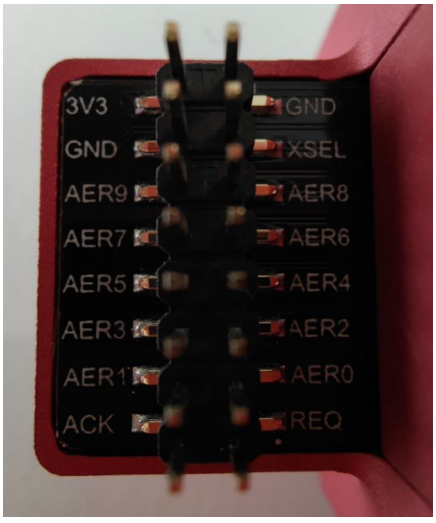
> **Warning**
>
> **The USB cable must always be connected for the device to work, even when using the AER interface.**

AER connector

The AER connector is located on the side, as shown in the following image:



It consists of 16 (2x8) 2.54mm pins. The exact pinout is marked on the connector itself, as shown here:



All data pins are 3.3V. A 3.3V reference and its Ground (GND) reference are provided on the connector. The 3.3V pin is only intended as a reference/output, do *not* power the device using this pin! The USB connector must be used always to provide power and configuration.

The protocol is documented *below*.

1.5.5 Optics

The camera lens mount is designed to accommodate CS-mount lenses. Other lenses can be accommodated by using adapters. The standard lens shipped with the camera is a C-mount lens and ships with an adapter. The chip requires a lens designed for 1/2" imagers.

The field of view (FOV) depends on the focal length **L** of the lens and the size **W** of the pixel array. It is computed from geometrical optics, not accounting for any lens distortion. The angular field of view (**AFOV**) is given by:

$$AFOV = 2 \arctan(\frac{W}{2L})$$

The linear FOV (**LFOV**) at a distance **D** from the lens is given by:

$$LFOV = \frac{D * W}{L}$$

The pixel array has a resolution of 346 x 260 and measures:

- Width: 346 pixels x 18.5 µm/pixel = 6.4 mm
- Height: 260 pixels x 18.5 µm/pixel = 4.81 mm
- Diagonal: 8 mm

Field of View Computations

The following table shows the horizontal and vertical field of view in degrees and its size at various distances for different common focal lengths.

Lens Focal Length [mm]	Horizontal Angular FoV [deg]	Vertical Angular FoV [deg]	Diagonal Angular FoV [deg]	Horizontal Linear FoV at 10 cm distance [cm]	Horizontal Linear FoV at 30 cm distance [cm]	Horizontal Linear FoV at 1 m distance [cm]	Horizontal Linear FoV at 2 m distance [cm]
2.10	113.46	97.75	124.64	30.48	91.44	304.81	609.62
2.80	97.64	81.32	110.06	22.86	68.58	228.61	457.21
3.00	93.70	77.44	106.31	21.34	64.01	213.37	426.73
3.60	83.28	67.49	96.07	17.78	53.34	177.81	355.61
4.50	70.84	56.24	83.32	14.22	42.67	142.24	284.49
6.00	56.15	43.69	67.43	10.67	32.01	106.68	213.37
9.00	39.15	29.92	47.96	7.11	21.34	71.12	142.24
12.00	29.87	22.67	36.90	5.33	16.00	53.34	106.68
16.00	22.62	17.10	28.10	4.00	12.00	40.01	80.01

1.5.6 Additional Information

Software

DAVIS 346 AER is compatible with all our software. You can use it in:

- *DV software*
- *dv-processing*

To enable the AER output connector, in DV open the `Capture` module full configuration (+ sign) and turn on the `ExternalAERControl` parameter.

For a low-level approach, there is also an example using libcaer to enable this feature on our developer website:

libcaer AER example[18]

AER protocol format

The AER protocol is a simple protocol using a variable number of lines (bus) to transmit data, and two lines (REQ and ACK) to synchronize the data between the sender and the receiver asynchronously using a four-phase handshake (this is also called a bundled asynchronous protocol). The ACK and REQ lines are active-low. This is how the protocol looks from the receiver's perspective, where REQ is to be considered an input and ACK an output:

[18] https://gitlab.com/inivation/dv/libcaer/-/blob/master/examples/davis_enable_aer.cpp

1. The receiver waits for the REQ line to be asserted by the sender
2. At this point, the data on the bus can be considered valid and stored
3. The receiver confirms having read the data by asserting ACK
4. It then waits until the sender has again deasserted REQ, deasserts ACK itself and goes back to wait in (1) for the next transaction

The following website has number of very detailed explanations for further reading:

https://www.cl.cam.ac.uk/~djg11/wwwhpr/fourphase/fourphase.html

Also, for details on AER, please look at:

https://www.ini.uzh.ch/~amw/scx/std002.pdf

For FPGA implementations, we recommend synchronizing at least the REQ input using a double-flip-flop synchronizer. Data itself should also synchronized in this way, or by connecting it directly to a register with an Enable signal and enabling it only during phase (2). All current iniVation DAVIS sensors employ a serial data format, meaning that the X and Y addresses are not output concurrently, but separately one after the other. One extra data bit, called XSelect, is used to disambiguate between the two types of address. Current sensors employ a row-wise readout scheme, so a Y (row) address will always be followed by a series of one or more X (column) addresses. The column address will also contain the Polarity information bit. Further, we recommend inserting a delay of ~50ns between REQ and getting the data (phases 1 and 2), because the current sensors violate the assumption that all data lines are always valid and stable before REQ is asserted. The XSelect bit can be considered valid right away, but for Y addresses this delay should be observed. X addresses can also be captured right away.

Note

DAVIS sensors may produce glitches known as "row-only events", where a Y (row) address is followed immediately by another Y (row) address. In this case, just discard the earlier address.

The format for DAVIS346 AER is documented in detail below:

```
AER bus width: 11 signals = 10 (9 downto 0) + XSelect

if XSelect = '1' then
  X Address, address is: 9 bits, 9 downto 1, polarity on 0
else
  Y Address, address is: 9 bits, 8 downto 0, 9 is don't care
end if;
```

FPGA receiver state-machine

An example VHDL state-machine for FPGAs is provided to interface with the AER connector and receive event data off the device:

VHDL state-machine[19]

Serial Number

The serial number of the device can be found on the case, usually a four-digit number printed on a black label located at the top of the camera case.

[19] https://docs.inivation.com/_static/code/davis346_aer_sm.vhd

Package Contents

DAVIS 346 AER ships with the following items:

- DAVIS 346 AER camera
- USB 3.0, 1m with locking screws
- Varifocal C mount lens (Datasheet[20])
- CS to C mount lens adapter
- Tripod

Safety Information

To prevent damage to property or injury to yourself or to others, read this safety information in its entirety before using this product.

- This product is intended to be used in a laboratory and for industrial applications under controlled conditions.
- We strongly recommend that you only use high quality USB cables, like the ones provided by iniVation. Using low quality USB cables could cause damages to the device.
- Keep the product dry. Do not handle the product with wet hands. Do not handle the plug with wet hands. Do not operate the camera near water. This could cause damage to the device. The camera is not water-safe.
- Handling: Handle your product with care. It is made of metal, glass, and plastic and has sensitive electronic components inside. The product can be damaged if dropped, burned, punctured, or crushed, or if it comes in contact with liquid. If you suspect damage to the product, please contact iniVation.
- Repairing: Do not open the product and do not attempt to repair the product yourself. Disassembling the product may damage it and will void your warranty. If your product is damaged or malfunctions, please contact iniVation.
- Do not disassemble or modify this product.
- Do not touch internal parts that become exposed as the result of a fall or other accident.
- Keep this product out of reach of children. Should a child swallow any part of this product, seek immediate medical attention.
- Use travel converters or adapters designed to convert from one voltage to another or with DC-to-AC inverters.
- Explosive and other atmospheric conditions. Connecting or using the product in any area with a potentially explosive atmosphere, such as areas where the air contains high levels of flammable chemicals, vapors, or particles (such as grain, dust, or metal powders), may be hazardous. Exposing the product to environments which have high concentrations of industrial chemicals, including near evaporating liquified gasses such as helium, may damage or impair the product's functionality.
- Turn this product off when its use is prohibited.
- Do not leave the product where it will be exposed to elevated temperatures for an extended period such as in an enclosed automobile or in direct sunlight. This can lead to malfunction.

[20] https://docs.inivation.com/_static/lenses/vari-focal-lens-incl.pdf

Correct Disposal





This product and its electronic accessories should not be disposed of with other household waste. If you are unable to dispose of this item safely please return it to iniVation AG.

CE Certification

DAVIS 346 AER is currently not certified for any purpose.

1.6 DVXplorer S Duo

Buy Device[21]

Download PDF[22]



Date: 2023-06-11

Note

This documentation is a work-in-progress. Please check back frequently for updates and provide us your feedback!

Please read carefully through the whole documentation to use the DVXplorer S Duo to its full potential and be aware of its limitations.

The DVXplorer S Duo (S standing for System) is an embedded camera, featuring a dual-sensor solution comprised of our DVS event-based sensor and a state-of-the-art OnSemi AR0234 global-shutter industrial color image sensor. Processing power is provided by an Nvidia Jetson Nano SOM[23], with a quad-core ARM Cortex-A57 processor, running at up to 1.43 GHz, 4 GB of DDR4 RAM and an Nvidia Maxwell GPU with 128 CUDA cores.

[21] https://inivation.com/buy/
[22] https://docs.inivation.com/_static/hardware_guides/dvxplorer-s-duo.pdf
[23] https://developer.nvidia.com/embedded/jetson-nano

1.6.1 Features

Hardware

- Nvidia Jetson Nano SOM (4x ARM Cortex-A57 @ 1.43 GHz, 4GB DDR4-RAM, Maxwell GPU @ 128 cores)
- iniVation DVS event sensor, 640x480 resolution (VGA)
- OnSemi AR0234 global-shutter color frame sensor, 1920x1200 resolution (WUXGA)
- Bosch BMI160 IMU sensor, 6 axis gyroscope + accelerometer
- 16 GB of eMMC storage total, 6 GB available for user data and recordings
- Gigabit Ethernet connector (RJ45), supports Power-Over-Ethernet (POE 802.3af)
- USB 3.0 Type C connector, with locking screws
- Mini-HDMI 1.4 connector
- 10x 2.54mm pins female connector for SPI, GPIO and PWM
- Size: H 32mm x W 80mm x D 92mm
- Weight: 220g, without lenses
- Lenses: 2x S-Mount (M12), 2x 6mm focal length provided with camera

Software

- Yocto Linux-based distribution
- Linux4Tegra kernel 4.19
- DV-SDK remote runtime
- Yocto-based SDK to cross-compile custom software
- Robust A/B update procedure, with separate partition for user data

1.6.2 Intended Usage

This device is meant to be used as an embedded platform for processing sensor data, running preferably attached to an Ethernet network.

It runs a custom Linux distribution, based upon the Yocto project[24], that directly integrates our DV-SDK software and automatically runs its runtime on boot. This allows users to connect using the GUI, setup modules and their connectivity on the embedded camera itself and run them there. Custom modules can also be uploaded and used.

Processing *performance is limited*, and will be at its worst if you try to use the camera like any of our other USB camera products, by just visualizing data on the DV GUI. Processing and data reduction should happen as much as possible on the device itself.

Data can be sent off the camera via either USB or Ethernet, but their *limited bandwidth* has to be taken into consideration. We recommend Ethernet for more robust deployments, due to its easier cabling situation and guaranteed higher power delivery capabilities.

1.6.3 Performance Considerations

Processing

The Nvidia Jetson Nano offers relatively limited CPU and GPU performance. Custom code will need to be carefully optimized to run in real-time.

[24] https://www.yoctoproject.org/

The amount of data that can be processed is limited and below what the sensors can provide at their peak performance.

The main bottlenecks for sensor data processing are:

- decoding of events from the camera into AEDAT4 format, caps at around 30 Mevents/s
- debayering (color interpolation) of image frames, caps at around 50 FPS

Todo

Later software releases will try to continuously improve the performance of algorithms and modules included by default in the DV-SDK software.

Networking

Getting data off the camera is limited by the respective networking connection's bandwidth:

- Ethernet: max. 1 Gbit/s
- USB 3.0 (network mode): max. 1.3 Gbit/s

A full resolution color frame at 1920x1200 takes 6.6 MB of data uncompressed, so you could transmit at most 18 frames per second, using all the bandwidth only for frames, over the Ethernet connection. Or, if sending only events, the USB connection would max out at around 9 Mevents/s. Sending all the data off the camera is *not* its intended mode of usage and will result in mediocre performance. It is imperative that as much processing and data reduction happens on the device. Best-case, only final features such as "person detected at position (X,Y)" or "42% activity in scene" would get sent off the device during normal operation.

The visualization function in the GUI also works by sending data out over the network and is constrained by these same limitations.

Compression can be enabled in the network output modules to improve the situation, though the trade-off is increased CPU usage. By default compression is disabled (**NONE**) for this reason, if you want to enable it, we recommend **LZ4**, as it is a good trade-off between compression and CPU usage.

Power

The Jetson Nano can run in two power states:

- 5W mode, plus system, total ~7.5W
- 10W MAXN mode, plus system, total ~12.5W

Higher power modes offer better CPU and GPU performance.

Ethernet POE 802.3af can provide up to 15W and thus run both modes without issues.

USB 3 can by default only provide up to 4.5W, so a normal USB 3.0 port will never be able to safely run this device. In truth, many USB ports and especially powered USB hubs can provide more power, often up to 15W (3A). In the case of USB Type A ports used with an A-to-C cable, this cannot be detected by the device. For USB Type C ports and cables, it is possible to detect higher power modes of 7.5W (1.5A) and 15W (3A) using the CC lines[25], if the host reports these properly.

For this reason, to try and guarantee power stability, our device always boots in the lowest power mode, and currently stays there.

[25] https://community.silabs.com/s/article/what-s-the-role-of-cc-pin-in-type-c-solution?language=en_US

Todo

Later software releases will detect and report as much information as possible concerning power available, mode selected and current power consumption. We'll also implement automatically switching to higher performance modes based upon that information, so as to always offer optimal performance. You can use the *nvpmodel* tool over SSH for now if you want to force higher performance modes manually.

Thermal

Warning

Please always be careful of the temperature when handling the device!

The device is passively cooled through the visible aluminium fins on its top. For best cooling performance, make sure these point up to allow for convective cooling to happen optimally. The device can become quite hot under continuous full load in the highest power settings, we have measured temperatures of up to 60°C (140° F) on the outer metal casing. Under normal load the temperature stays around 40° C (104° F).

Todo

Later software releases will report in detail on measured temperature. You can use the *tegra-stats* tool over SSH for now if you want to get this information displayed.

1.6.4 Time Synchronization

The timestamps of event, frame and IMU data in DV-SDK are defined as microseconds since the Unix Time Epoch[26]. Since the device runs Linux, it must be able to accurately set its real-time clock and keep it synchronized. This is a well-known and well-solved problem using network time synchronization techniques, such as the Network Time Protocol (NTP) or the Precision Time Protocol (PTP). It is even possible to use external hardware such as GPS and Atomic Clocks for certain use cases. For network time synchronization to work, the device must be able to reach an external network, usually the internet, and talk to pool.ntp.org[27]. If you connect over Ethernet with DHCP, we assume the resulting connection can access the internet and the global NTP servers. If you connect over USB, you'll have to enable network forwarding for the camera first, *see below* for instructions.

Todo

Support for setting NTP server addresses manually and getting them from DHCP will come in a later software release. Later software releases will also show the time synchronization status in an easy to digest manner in DV.

Note

Support for advanced time synchronization via PTP and GPS is not supported on the current hardware.

The network chip of the Nvidia Jetson Nano does not support PTP hardware timestamping, nor is it possible to attach most common GPS receivers since the UART pins are not exposed.

[26] https://en.wikipedia.org/wiki/Unix_time
[27] https://www.ntppool.org/en/

USB Network Forwarding

Network forwarding enables the device to access the wider network and internet when connected over USB. This is required for time synchronization features, and in the future, for automatic system image updates.

Linux

Execute the following shell-script as root to enable network forwarding to the camera:

```
sudo ./usb-internet.sh
```

Script content (*usb-internet.sh*):

```
#!/bin/sh

echo 1 > /proc/sys/net/ipv4/ip_forward
iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -j ACCEPT -s 192.168.55.0/24
iptables -A POSTROUTING -t nat -j MASQUERADE -s 192.168.55.0/24
```

MacOS

Todo

MacOS USB Network Forwarding instructions not yet available.

Windows

Todo

Windows USB Network Forwarding instructions not yet available.

1.6.5 DV-runtime access

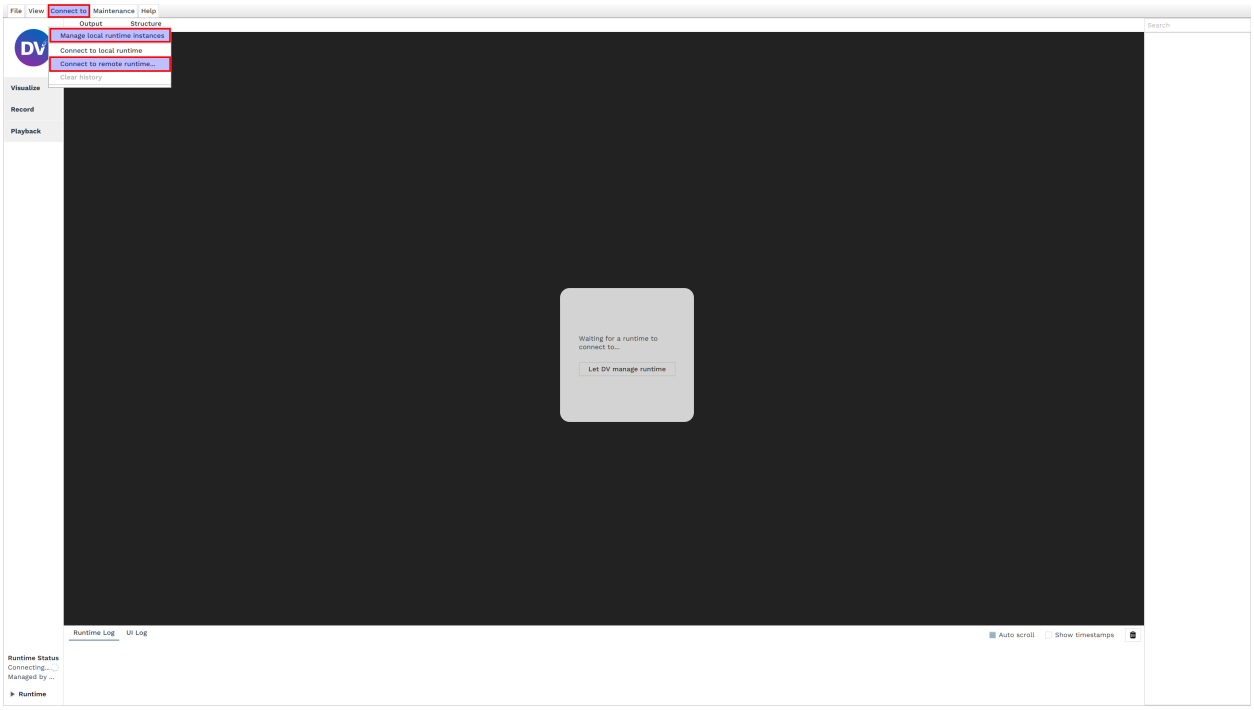
The embedded camera is running an instance of our DV-SDK software's runtime. It allows to connect to the camera from the GUI remotely, visualize data, add/remove and connect modules to process data.

Via USB

Open the GUI, in the top menu select "Connect to", in the resulting drop-down menu disable "Manage local runtime instances", the greyed out "Connect to remote runtime…" will then become available. Select it and in the ensuing pop-up enter the following parameters:

- IP Address: 192.168.55.1
- Port: 4040

Finally, click on Connect.



Note

When connecting via USB, the IP address is always this same one. As such, only *one* camera can be connected via USB at the same time to the same host system.

Via Ethernet

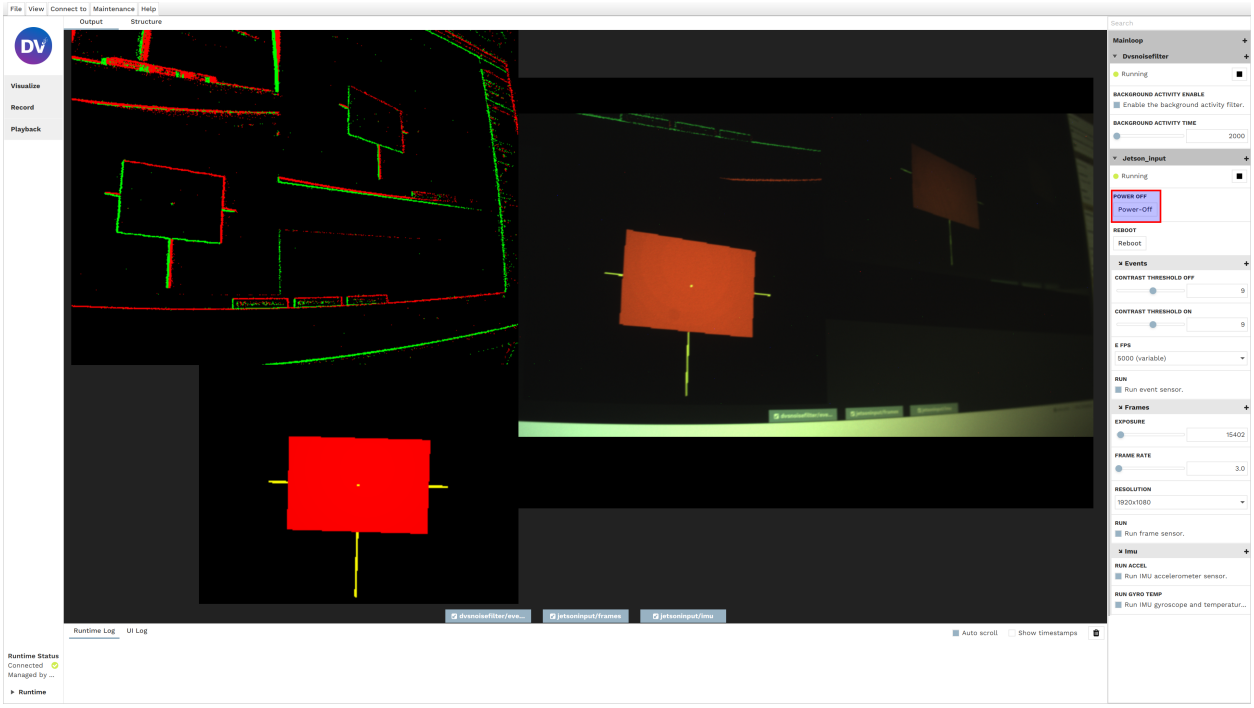
The same procedure as for USB applies to connect to the remote runtime, with the difference the IP address will be different and depend on your network configuration. Usually DHCP will assign an unique IP to the camera automatically. The presence of the camera's dv-runtime service on the network is advertised through the mDNS auto-discovery service, you can use tools such as *avahi-discover* to see all the cameras on your network. Alternatively ask your network administrator to check and communicate the address to you.

Todo

Later software releases will allow more flexible network configurations. For now, if you have need for a very specific network configuration, SSH into the device and edit the systemd-networkd configuration file at */data/config/20-wired.conf* manually.

1.6.6 DV-runtime usage

The default configuration for the DV-SDK runtime always contains a *dv_jetson_input* module, which is the interface to the system's three main sensors.



All sensor settings can be reached from the right side-menu, as usual click on the + sign to access and search the full list of options. The most important options are already displayed in the right side-menu by default.

Power-Off / Reboot

Warning

Please power off the device using the appropriate *Power-Off* button in DV!

Of particular note are the *Power-Off* and *Reboot* buttons. As the names suggest, they turn off or reboot the device. It is very important to properly power off the device through clicking this button or, alternatively, sending the `poweroff` command via SSH. Improperly shutting the device down by disconnecting its power supply (USB or Ethernet cable) will result in the loss of the current DV runtime configuration, as it is only saved at shut-down. Further, if any recording was happening at the time (*dv_output_file* module), it will be truncated and improperly terminated, which can lead to loss of data and performance.

Todo

Later software releases will offer a button to manually save the current configuration to the device at any point in time. It will still be recommended to properly shut it down via the appropriate buttons.

You can also always use the DV GUI Projects feature to save and reload any configuration states as XML files.

Frame Sensor Control

The most important controls for the frame sensor concern the frame-rate, the exposure and the resolution. The resolution defines the number of pixels to read out, which takes a certain amount of time *T_READOUT* for a specific resolution. This defines the maximum theoretical frame-rate. The frame-rate control allows the artifical lengthening of *T_READOUT* to slow down readout to a preferred rate by the user, let's call that time *T_READOUT_USER*. The *MAXIMUM(T_READ-*

OUT_USER, EXPOSURE) defines the effective time needed to read a frame, and thus the real frame-rate you'll get from the device.

Todo

Later software releases will introduce automatic exposure control support, as well as more fine-grained controls for gain and other advanced features.

1.6.7 SSH access

Warning

Please set a secure root password for SSH access as soon as possible!

It is possible to access the device via SSH as root, currently no password is set by default. We urge users to set a secure password as soon as possible. SFTP is also enabled to upload custom DV modules and update the system image.

- IP Address: see *DV-runtime access*
- Port: 22
- User: root
- Password: *none*

Todo

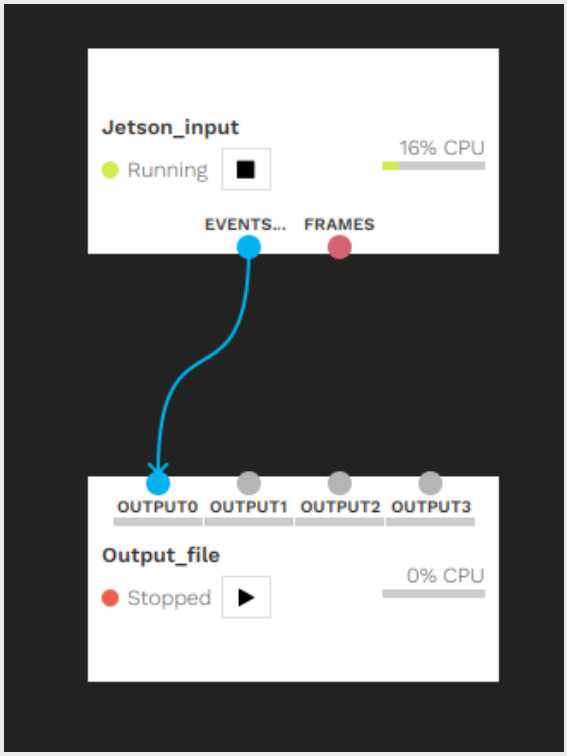
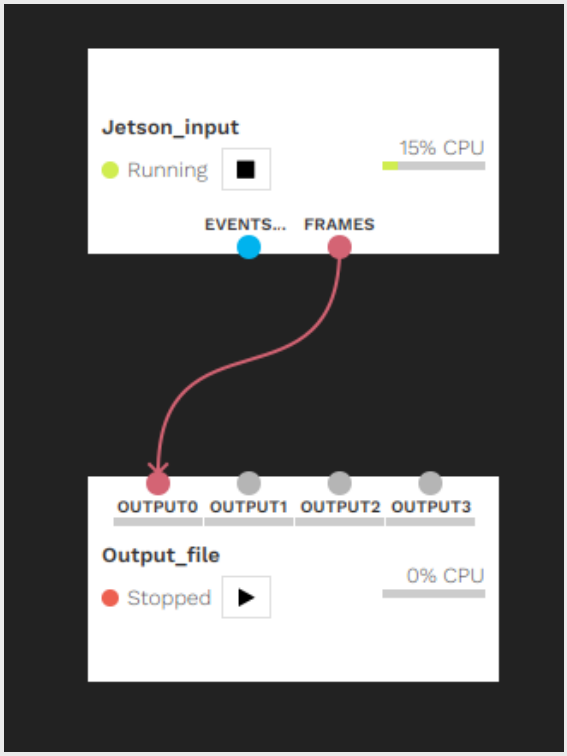
Future software releases will change how authentication works, disabling password-less root access and requiring users to log-in using SSH keys. A method to upload/import SSH keys over DV will be provided.

1.6.8 Calibration

At present, there is no dedicated calibration support available. Users are required to carry out standard camera calibration themselves. Nevertheless, due to the limitations of the current hardware, it is not feasible to execute the calibration procedure directly on the DVXplorer-S DUO, as it cannot handle the computational demands.

In order to calibrate with the help of an external pc or server, follow these steps:

1. Connect the camera to your computer *via USB* or *via Ethernet*.
2. Open DV software and connect to the camera runtime as explained *here*. This will allow you to visualize the camera output. Use the gui to:
3. Focus the camera using *these instructions*.
4. Record raw event data (or frames for the image sensor) from the camera while moving the calibration pattern in front of it. You can do this by setting up the proper structure in the camera runtime (connect "jetson input" to "dv output file" module), as follows:

Record Events Only	Record Frames Only
	

Note: The recording will be stored on the camera.

5. Once the recording is finished: on your computer, copy it from the camera using:

```
scp root@<your_camera_address>:/var/lib/dv-runtime/<your_recording.aedat4>
↪<local_directory>
```

6. Disconnect DV from remote runtime.
7. Connect DV to local runtime and run the calibration procedure following *these instructions*. Remember to replace "camera input" with "input_file" module, and select the previously downloaded aedat4 file (*/local/path/to/<your_recording.aedat4>*) as input.

1.6.9 Write and compile your own modules

You can write your own DV-SDK C/C++ modules to process data following your personal requirements. You will need to compile them specifically to run on the DVXplorer S Duo. To do this, you'll need to install the SDK and configure your development environment to use it for building your module. We currently provide detailed instructions for the JetBrains CLion IDE[28].

Todo

Instructions for setting up the Yocto SDK with the Qt-Creator and Visual Studio Code IDEs will follow.

[28] https://www.jetbrains.com/clion/

Todo

Support for installing the SDK on Windows and MacOS will follow. MacOS should work the same as Linux but has not yet been tested.

Install Yocto SDK on Linux

Download the latest Yocto SDK installer for Linux (.sh file) from release.inivation.com[29].

Latest official release[30] (2023-06-07_13-18-51 git revision ef6d1e5c)

Install the SDK by executing the installer:

```
chmod 0755 ./inivation-image-full-jetson-nano-inivation-v2-x86_64-glibc-toolchain-
↪2023-06-07_13-18-51-master-ef6d1e5c.sh
./inivation-image-full-jetson-nano-inivation-v2-x86_64-glibc-toolchain-2023-06-07_13-
↪18-51-master-ef6d1e5c.sh
```

Install it to a user-accessible, easy to remember folder like */home/youruser/inivation-sdk/*.

Note

If you're not running the latest system image, you'll have to download and install the correct version of the SDK for that specific version of the system image. To find out the system image version you're running, SSH into the device and take a look at the */etc/build* file.

Todo

Later software versions will show build version information in DV.

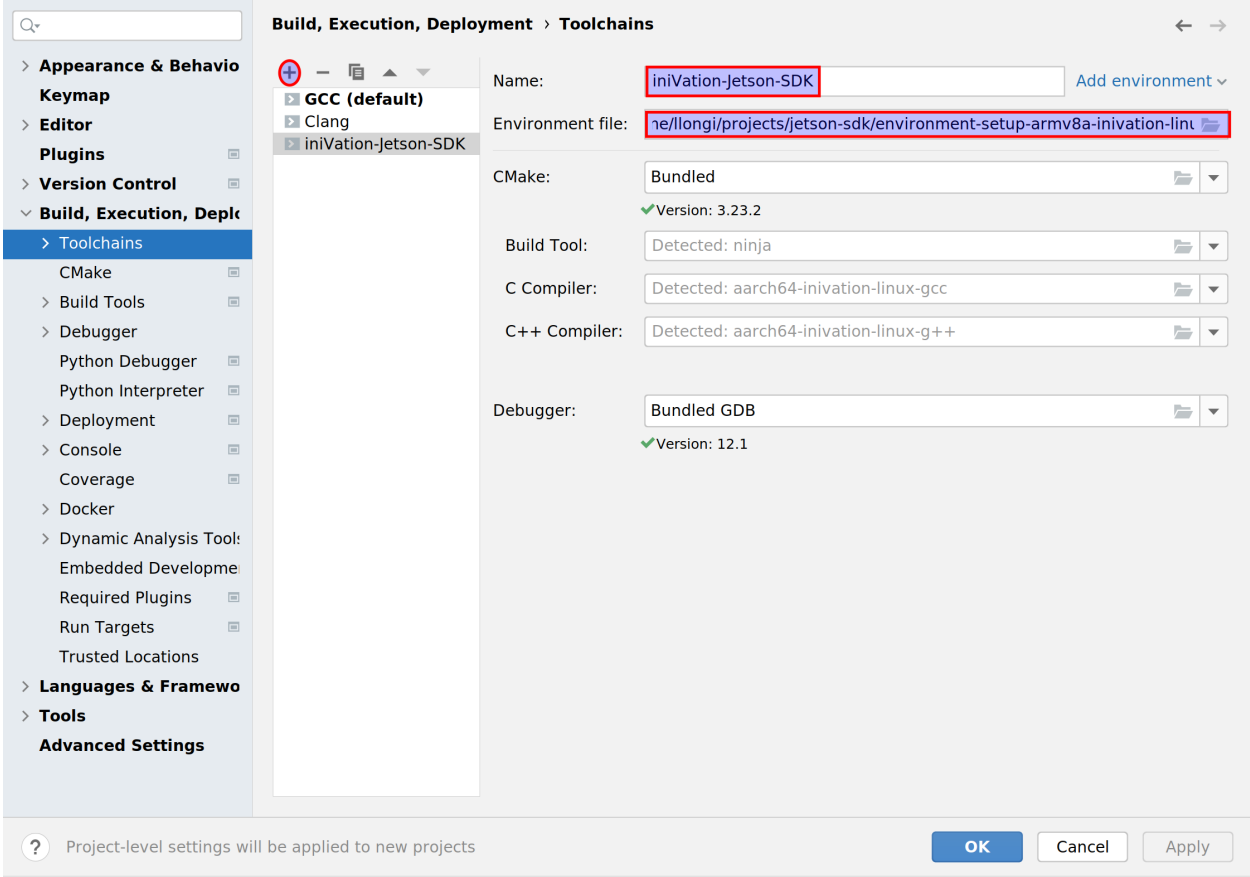
Add Yocto SDK to CLion

Make sure to have at least version 2023.1 of CLion installed. In CLion you first have to add the SDK as a toolchain. Go to "Settings -> Build, Execution, Deployment -> Toolchains", press the + sign above the left-side list of toolchains and then select "System" to add a new system toolchain. Now change its name to "iniVation-Jetson-SDK" and click on "Add environment -> From File", then in the "Environment file" dialog select the SDK environment file at:

```
/home/youruser/inivation-sdk/environment-setup-armv8a-inivation-linux
```

[29] https://release.inivation.com/?prefix=jetson/jetson-nano-inivation-v2/master/

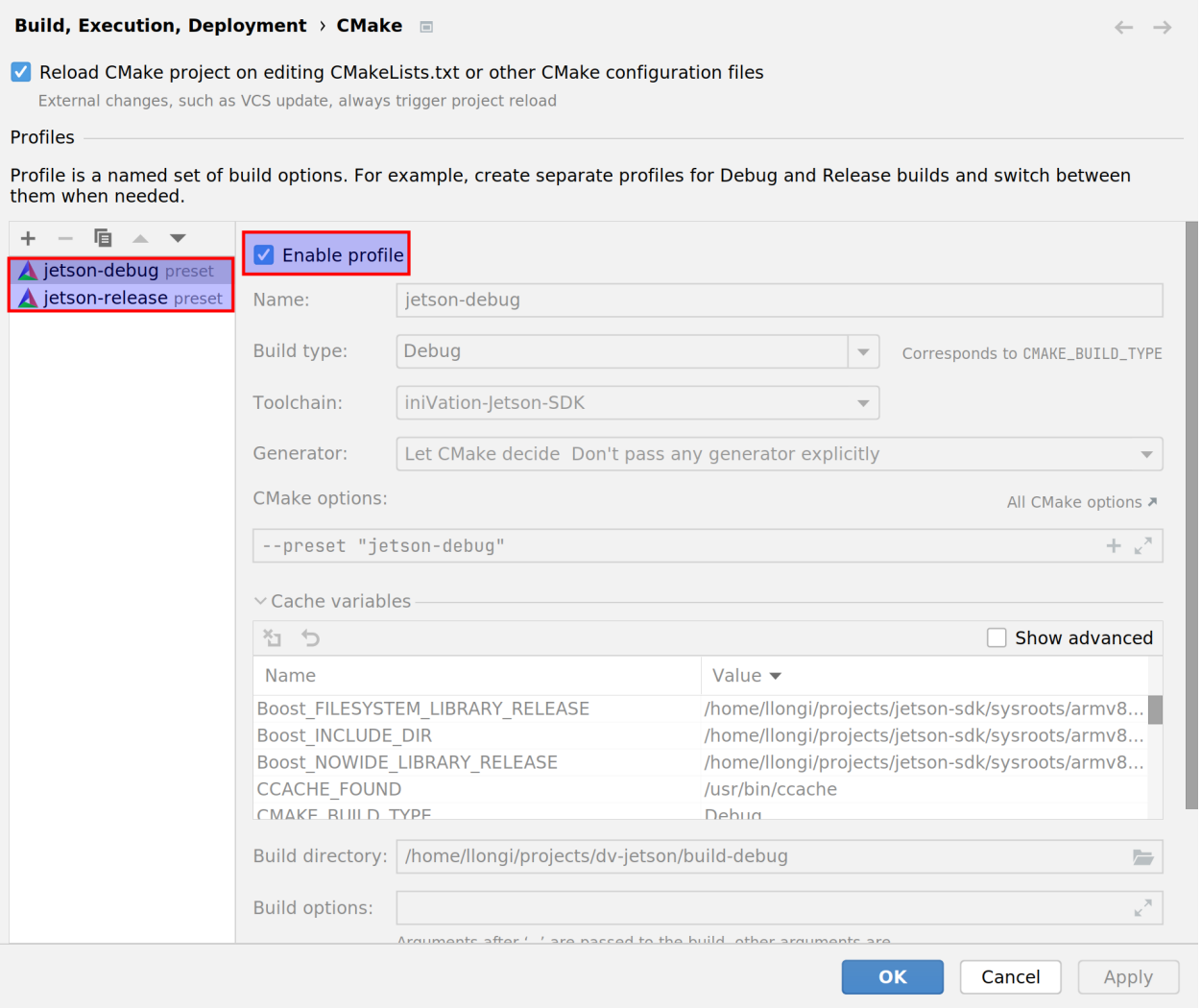
[30] https://s3.eu-central-1.amazonaws.com/release.inivation.com/jetson/jetson-nano-inivation-v2/master/inivation-image-full-jetson-nano-inivation-v2-x86_64-glibc-toolchain-2023-06-07_13-18-51-master-ef6d1e5c.sh



Leave everything else unchanged, CMake and Debugger should remain the "Bundled" version. Please ensure the name is exactly "**iniVation-Jetson-SDK**".

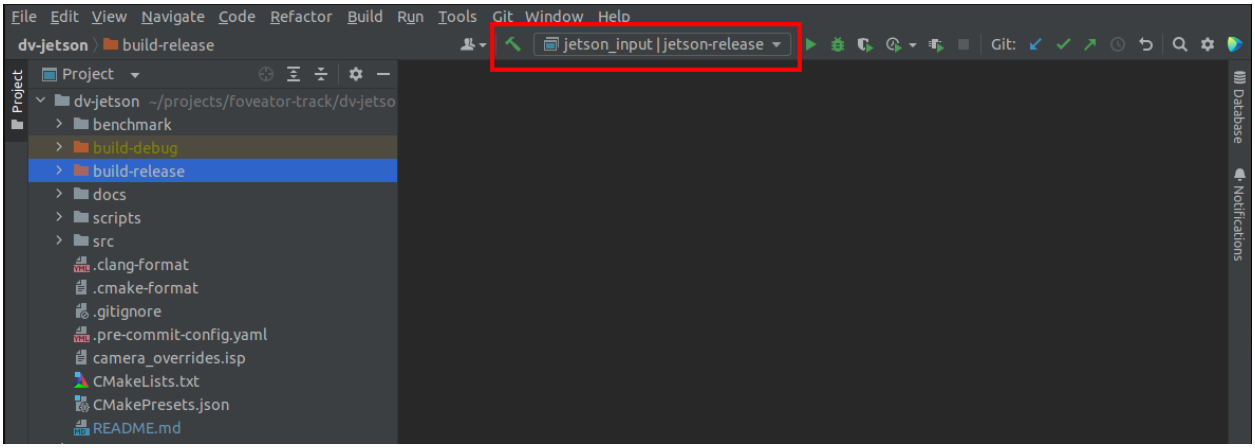
Open Project in CLion

Now open the *dv-example-cpp* project as you would normally by navigating to it and selectings its directory. The CMake Configuration window will pop up, disable and remove the default 'Debug' or 'Release' profiles, and instead enable the 'jetson-debug' and 'jetson-release' profiles. These are automatic CMake Presets that will configure all the needed paths and variables for you.



Build a Module in CLion

Once the CMake presets are set, you can run CMake and build the modules as you would normally do using CLion.



Transfer Modules to Device

To ease the transfer of built modules to the camera device during testing, we recommend using a script to automate this operation. The following will move the module to its proper place and restart dv-runtime on the device so it gets reloaded.

```
Usage: ./module-to-system.sh <device-IP-address> <path-to-built-module>

./module-to-system.sh 192.168.55.1 build-release/user_example.so
```

Script content (*module-to-system.sh*):

```
#!/bin/sh

sftp root@$1:/var/lib/dv-runtime/modules <<< $"put $2"
ssh root@$1 "systemctl restart dv-runtime.service"
```

1.6.10 Update system image

Download the latest system image from release.inivation.com[31], you will need the *.mender* file.

Latest official release[32] (2023-06-07_13-18-51 git revision ef6d1e5c)

Upload it to the device using SFTP, we recommend putting the file in the temporary directory */var/tmp/*:

```
sftp root@192.168.55.1:/var/tmp/ <<< "put inivation-image-full-jetson-nano-inivation-
↪v2-2023-06-07_13-18-51-master-ef6d1e5c.mender"
```

Then log-in to the device via SSH, execute the installation command and reboot:

```
ssh root@192.168.55.1
mender install /var/tmp/inivation-image-full-jetson-nano-inivation-v2-2023-06-07_13-
↪18-51-master-ef6d1e5c.mender
reboot
```

Once the device has rebooted, if everything appears to work correctly, log-in via SSH again and confirm the update:

```
ssh root@192.168.55.1
mender commit
```

In case you forget this last step, the device will automatically roll-back to the previously installed version on the next reboot. This is not an issue, you can simply execute the upgrade steps again.

Todo

Later software releases will implement automatic commit upon successfull update completion, as well as automatic discovery, download and installation of updates on users' request.

[31] https://release.inivation.com/?prefix=jetson/jetson-nano-inivation-v2/master/

[32] https://s3.eu-central-1.amazonaws.com/release.inivation.com/jetson/jetson-nano-inivation-v2/master/inivation-image-full-jetson-nano-inivation-v2-2023-06-07_13-18-51-master-ef6d1e5c.mender

1.6.11 History

- 2023-06-11: first version of documentation published

1.6.12 Open Issues

Todo

Later software releases will try to continuously improve the performance of algorithms and modules included by default in the DV-SDK software.

original entry

Todo

Later software releases will detect and report as much information as possible concerning power available, mode selected and current power consumption. We'll also implement automatically switching to higher performance modes based upon that information, so as to always offer optimal performance. You can use the *nvpmodel* tool over SSH for now if you want to force higher performance modes manually.

original entry

Todo

Later software releases will report in detail on measured temperature. You can use the *tegra-stats* tool over SSH for now if you want to get this information displayed.

original entry

Todo

Support for setting NTP server addresses manually and getting them from DHCP will come in a later software release. Later software releases will also show the time synchronization status in an easy to digest manner in DV.

original entry

Todo

MacOS USB Network Forwarding instructions not yet available.

original entry

Todo

Windows USB Network Forwarding instructions not yet available.

original entry

Todo

Later software releases will allow more flexible network configurations. For now, if you have need for a very specific network configuration, SSH into the device and edit the systemd-networkd configuration file at */data/config/20-wired.conf* manually.

original entry

Todo

Later software releases will offer a button to manually save the current configuration to the device at any point in time. It will still be recommended to properly shut it down via the appropriate buttons.

You can also always use the DV GUI Projects feature to save and reload any configuration states as XML files.

original entry

Todo

Later software releases will introduce automatic exposure control support, as well as more fine-grained controls for gain and other advanced features.

original entry

Todo

Future software releases will change how authentication works, disabling password-less root access and requiring users to log-in using SSH keys. A method to upload/import SSH keys over DV will be provided.

original entry

Todo

Instructions for setting up the Yocto SDK with the Qt-Creator and Visual Studio Code IDEs will follow.

original entry

Todo

Support for installing the SDK on Windows and MacOS will follow. MacOS should work the same as Linux but has not yet been tested.

original entry

Todo

Later software versions will show build version information in DV.

original entry

Todo

Later software releases will implement automatic commit upon successfull update completion, as well as automatic discovery, download and installation of updates on users' request.

original entry

Safety Information

To prevent damage to property or injury to yourself or to others, read this safety information in its entirety before using this product.

- This product is intended to be used in a laboratory and for industrial applications under controlled conditions.
- We strongly recommend that you only use high quality USB cables, like the ones provided by iniVation. Using low quality USB cables could cause damages to the device.
- Keep the product dry. Do not handle the product with wet hands. Do not handle the plug with wet hands. Do not operate the camera near water. This could cause damage to the device. The camera is not water-safe.
- Handling: Handle your product with care. It is made of metal, glass, and plastic and has sensitive electronic components inside. The product can be damaged if dropped, burned, punctured, or crushed, or if it comes in contact with liquid. If you suspect damage to the product, please contact iniVation.
- Repairing: Do not open the product and do not attempt to repair the product yourself. Disassembling the product may damage it and will void your warranty. If your product is damaged or malfunctions, please contact iniVation.
- Do not disassemble or modify this product.
- Do not touch internal parts that become exposed as the result of a fall or other accident.
- Keep this product out of reach of children. Should a child swallow any part of this product, seek immediate medical attention.
- Use travel converters or adapters designed to convert from one voltage to another or with DC-to-AC inverters.
- Explosive and other atmospheric conditions. Connecting or using the product in any area with a potentially explosive atmosphere, such as areas where the air contains high levels of flammable chemicals, vapors, or particles (such as grain, dust, or metal powders), may be hazardous. Exposing the product to environments which have high concentrations of industrial chemicals, including near evaporating liquified gasses such as helium, may damage or impair the product's functionality.
- Turn this product off when its use is prohibited.
- Do not leave the product where it will be exposed to elevated temperatures for an extended period such as in an enclosed automobile or in direct sunlight. This can lead to malfunction.

Correct Disposal





This product and its electronic accessories should not be disposed of with other household waste. If you are unable to dispose of this item safely please return it to iniVation AG.

1.6.13 Certifications

DVXplorer S Duo is currently not certified for any purpose.

1.7 Stereo Kit

Buy Device[33]

Download PDF[34]



We offer stereo kit for advanced users who wants to explore stereo vision. The stereo kit includes two cameras of choice and accompanying utilities.

1.7.1 Package Contents

The Stereo kit ships with the following items:

- 2 x iniVation device (*DAVIS346*, *DVXplorer* or *DVXplorer Lite*
- 2 x Micro-USB 3.0 cable
- 2 x Varifocal C mount lens (Datasheet[35])

[33] https://inivation.com/buy/
[34] https://docs.inivation.com/_static/hardware_guides/stereo-kit.pdf
[35] https://docs.inivation.com/_static/lenses/vari-focal-lens-incl.pdf

- 2 x Tripod
- 2 x Carry case
- 2 x Stereo plate
- 1 x Sync cable

1.7.2 Safety Information

To prevent damage to property or injury to yourself or to others, read this safety information in its entirety before using this product.

- This product is intended to be used in a laboratory and for industrial applications under controlled conditions.
- We strongly recommend that you only use high quality USB cables, like the ones provided by iniVation. Using low quality USB cables could cause damages to the device.
- Keep the product dry. Do not handle the product with wet hands. Do not handle the plug with wet hands. Do not operate the camera near water. This could cause damage to the device. The camera is not water-safe.
- Handling: Handle your product with care. It is made of metal, glass, and plastic and has sensitive electronic components inside. The product can be damaged if dropped, burned, punctured, or crushed, or if it comes in contact with liquid. If you suspect damage to the product, please contact iniVation.
- Repairing: Do not open the product and do not attempt to repair the product yourself. Disassembling the product may damage it and will void your warranty. If your product is damaged or malfunctions, please contact iniVation.
- Do not disassemble or modify this product.
- Do not touch internal parts that become exposed as the result of a fall or other accident.
- Keep this product out of reach of children. Should a child swallow any part of this product, seek immediate medical attention.
- Use travel converters or adapters designed to convert from one voltage to another or with DC-to-AC inverters.
- Explosive and other atmospheric conditions. Connecting or using the product in any area with a potentially explosive atmosphere, such as areas where the air contains high levels of flammable chemicals, vapors, or particles (such as grain, dust, or metal powders), may be hazardous. Exposing the product to environments which have high concentrations of industrial chemicals, including near evaporating liquified gasses such as helium, may damage or impair the product's functionality.
- Turn this product off when its use is prohibited.
- Do not leave the product where it will be exposed to elevated temperatures for an extended period such as in an enclosed automobile or in direct sunlight. This can lead to malfunction.

1.7.3 Correct Disposal





This product and its electronic accessories should not be disposed of with other household waste. If you are unable to dispose of this item safely please return it to iniVation AG.

To purchase or to get a detailed comparison of our products, please visit our Buy Page[36].

[36] https://inivation.com/buy/

Device	Appearance
DVXplorer	
DVXplorer Lite	
DVXplorer Micro	
DAVIS346	
DAVIS346 AER	
DVXplorer S Duo	
Stereo Kit	

CHAPTER

TWO

DISCONTINUED PRODUCTS

Device	Appearance	User Guide
DVXplorer Mini		Download PDF[37]
DVS240		Download PDF[38]
DAVIS240		Download PDF[39]
eDVS		Download PDF[40]
DVS128		Download PDF[41]
DVL-5000		User Guide

[37] https://docs.inivation.com/_static/hardware_guides/dvxplorer-mini.pdf
[38] https://docs.inivation.com/_static/hardware_guides/dvs240.pdf
[39] https://docs.inivation.com/_static/hardware_guides/davis240.pdf
[40] https://docs.inivation.com/_static/hardware_guides/edvs.pdf
[41] https://docs.inivation.com/_static/hardware_guides/dvs128.pdf

CHAPTER

THREE

ADVANCED USAGE

- We published a *white paper* on *Understanding the Performance of Neuromorphic Event-based Vision Sensors*.
- Each one of our camera models has its unique firmware that can and should *be updated* from time to time.
- Documentation on how to install the USB drivers to use our devices can be found *here*
- More information on IMU can be found *here*.
- In a lot of scenarios, proper usage of our cameras depend on the bias settings being properly set. Follow *this guide* to understand them more in details.
- To use our cameras in a stereo setup with another one of our cameras or an external one, we provide *camera synchronization through sync connection*.

3.1 White Paper

This white paper (Download PDF[42]) deals with *Understanding the Performance of Neuromorphic Event-based Vision Sensors*, in particular temporal resolution, synchronous versus asynchronous readout, dynamic range and low light performance.

3.2 Firmware Update

Warning

Please do not update the firmware unless one of our tools explicitly tells you to do so!

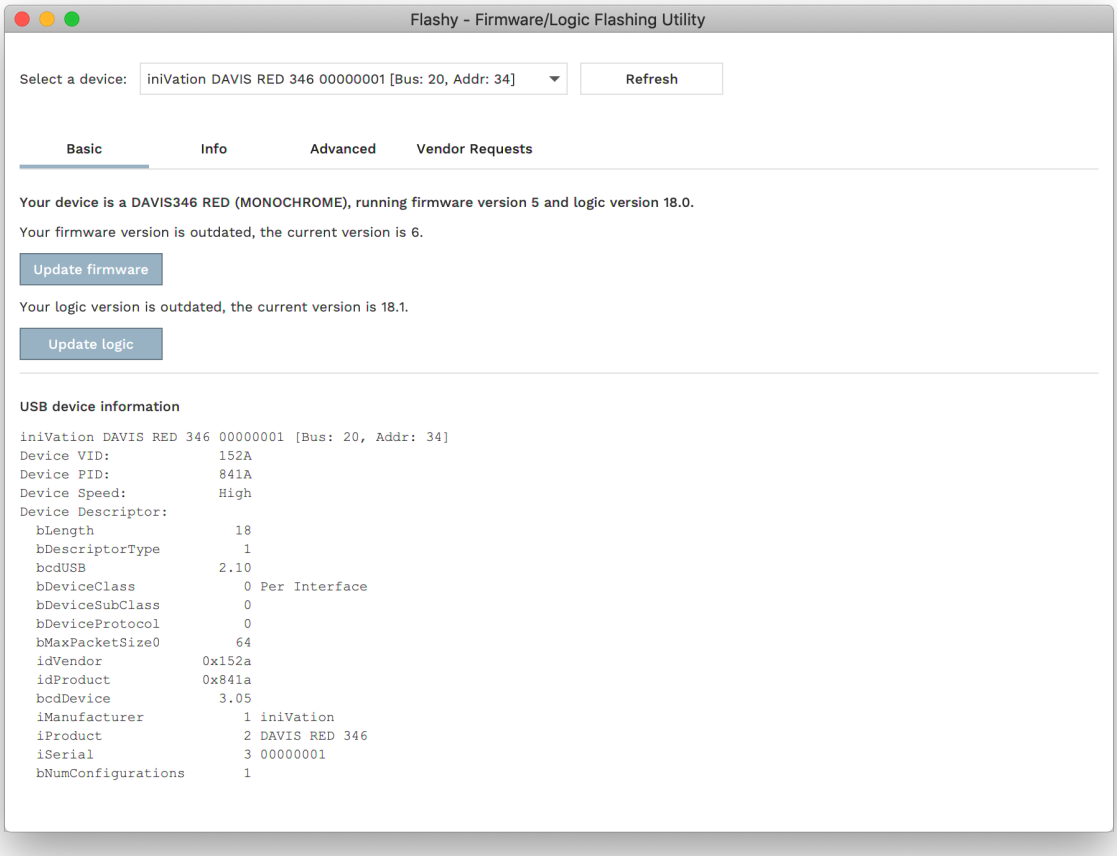
Always use the latest version of DV and the Flashy tool to perform a firmware update!

If our software cannot access the device, the problem is often not the firmware or logic versions, but USB permissions or drivers. Also the incorrect usage of the DV GUI, by not selecting the device from the drop-down menu, often leads to issues. Please visit our *FAQ* for more details.

The software running on the camera can be updated as follows:

1. Open *DV*
2. Select *Maintenance -> Update camera firmware*
3. Wait for the firmware update window to appear

[42] https://docs.inivation.com/_static/white_papers/White-Paper-May-2020.pdf



4. If there is an update available, the tool will display the update. If the newest firmware version is not listed, you may want to update DV
5. If available, click *Update firmware* and *Update logic*
6. **Unplug and re-plug the camera**

3.3 USB Device Installation

Most of our devices use USB to communicate with a host system. To ensure our software can recognize and interact with them, the correct drivers have to be installed and the correct permissions granted, depending on the operating system in use.

3.3.1 Linux

Access to USB devices under Linux is provided by the standard kernel USB drivers and the libusb library. Permissions for the user running our software have to be configured correctly to access the device. You must grant the user access to the USB device. The appropriate udev files must be installed. Our libcaer packages for Arch, Fedora, Ubuntu and Gentoo already provide these files automatically, and libcaer compiled from source will also try to install them into the appropriate location at /lib/udev/rules.d/. If you're not using libcaer, or want to add the files manually, this can be achieved by creating, as root, the appropriate udev rules files:

```
/etc/udev/rules.d/65-inivation.rules

/etc/udev/rules.d/66-inivation_dev.rules
```

You can find ready-to-use udev rules files in our Git repository[43].

If you're using a distribution that supports SELinux tags (i.e. Fedora since Fedora Core 2; Debian as of the Etch release; Ubuntu as of 8.04 Hardy Heron; OpenSUSE contains SELinux "basic enablement" as of version 11.1; SUSE Linux Enterprise 11 features SELinux as a "technology preview"), please use the udev rules files in the selinux/ sub-folder.

```
udevadm control --reload-rules
```

or, for newer udev versions:

```
udevadm control --reload
```

Now unplug and replug the camera into your computer.

3.3.2 MacOS X

No particular steps have to be performed on MacOS X to access our USB devices. Simply connect them to your system and start working with them!

3.3.3 Windows

On Windows, our USB devices use the standard WinUSB driver. All DAVIS models automatically request this driver from Windows the first time they are connected, and if Windows already has the driver in its database or has Automatic Driver Update activated, it should just find the driver for you and properly install it. Some times this doesn't happen, or the currently installed version is either too old or incompatible, in that case you'll have to install the driver manually using the Zadig tool. Very early DVS128 users (pre-2014) might also still have the old Thesycon UsbIO driver installed; this must be substituted with the WinUSB driver manually. Please read the following section of this manual for more detailed instructions.

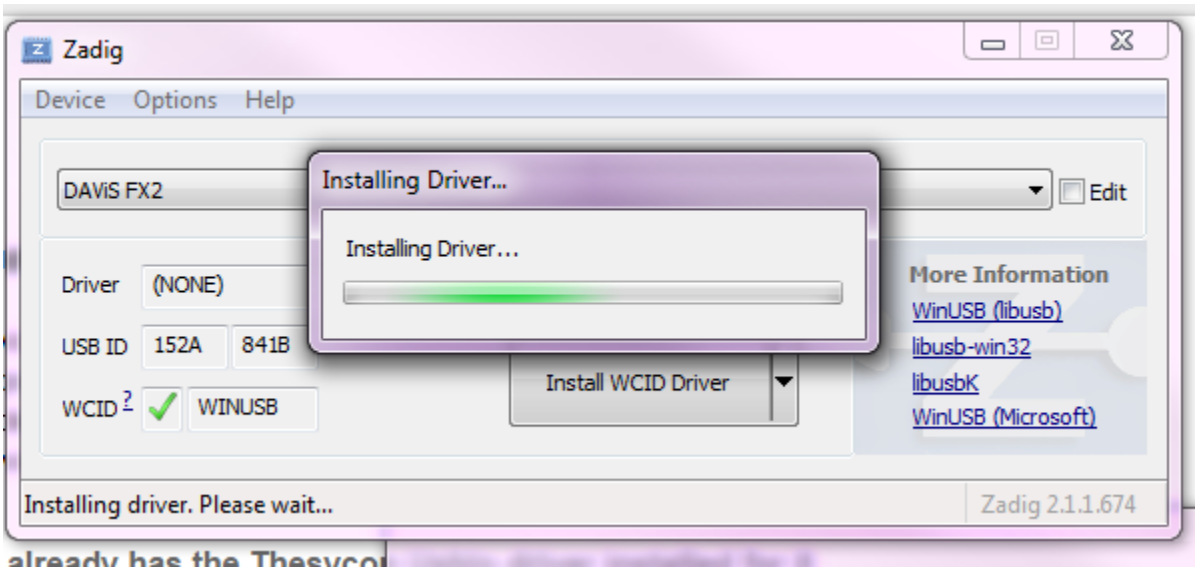
Using Zadig to install the WinUSB driver

Zadig is available from its official website[44].

Once you start Zadig, you should see a list of devices. If not, go to options and tick "List all devices". Make sure you choose the correct device (not for instance your mouse!).

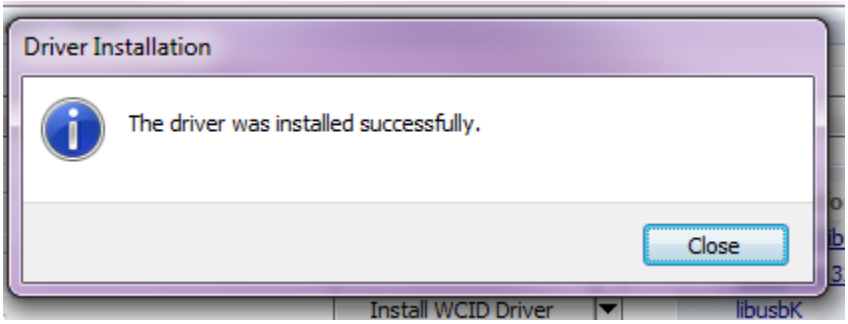
Then click *Install WCID Driver* to install the WinUSB driver: (WCID devices[45] are installed automatically for new instances of devices plugged into the computer)

[43] https://gitlab.com/inivation/hardware/devices-bin/tree/master/drivers/linux/udev-rules
[44] http://zadig.akeo.ie/
[45] https://github.com/pbatard/libwdi/wiki/WCID-Devices



If you already have a device installed, you may instead need to select *Replace Driver* rather than *Install WCID Driver*. You should check after installation (see below) that you actually have the correct WinUSB driver installed.

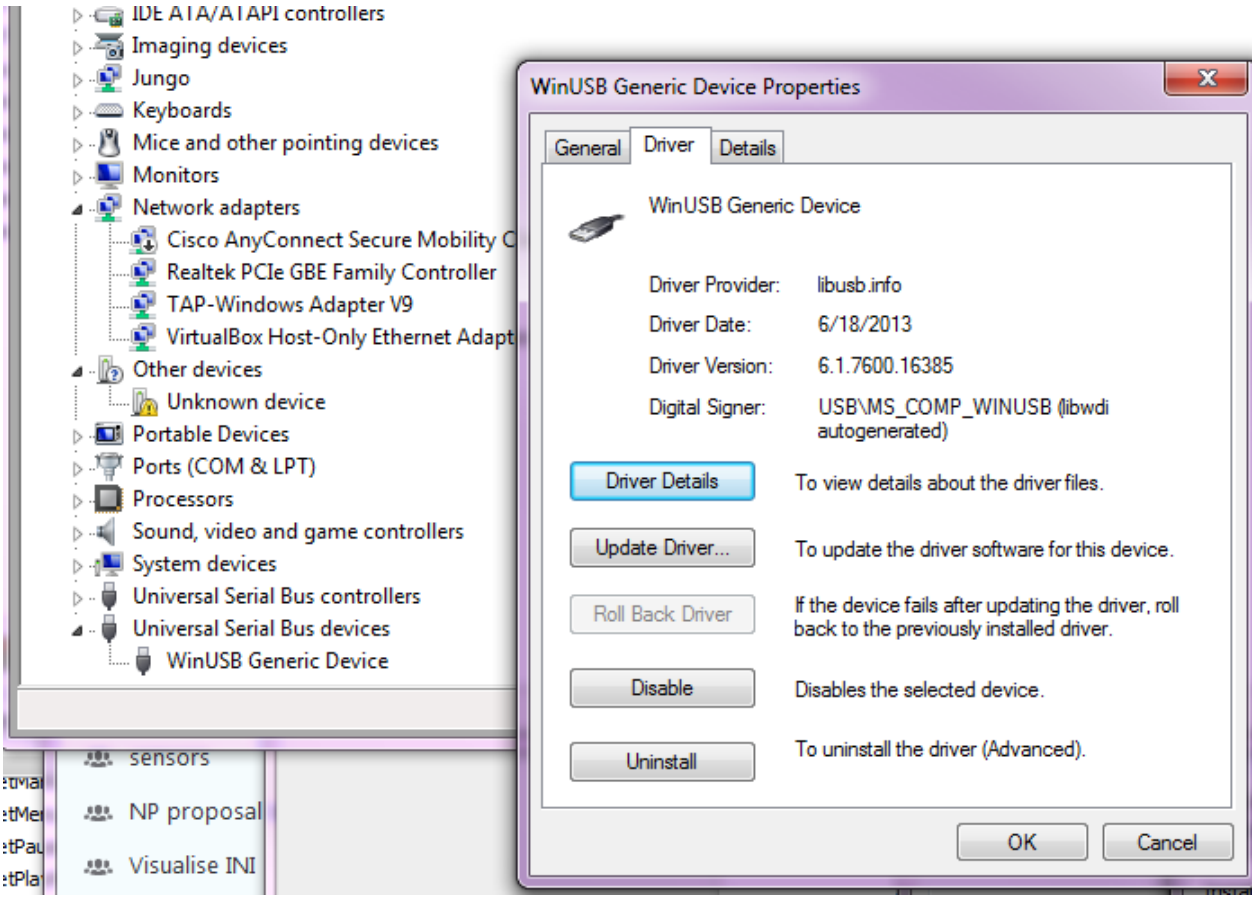
You will be notified once done.



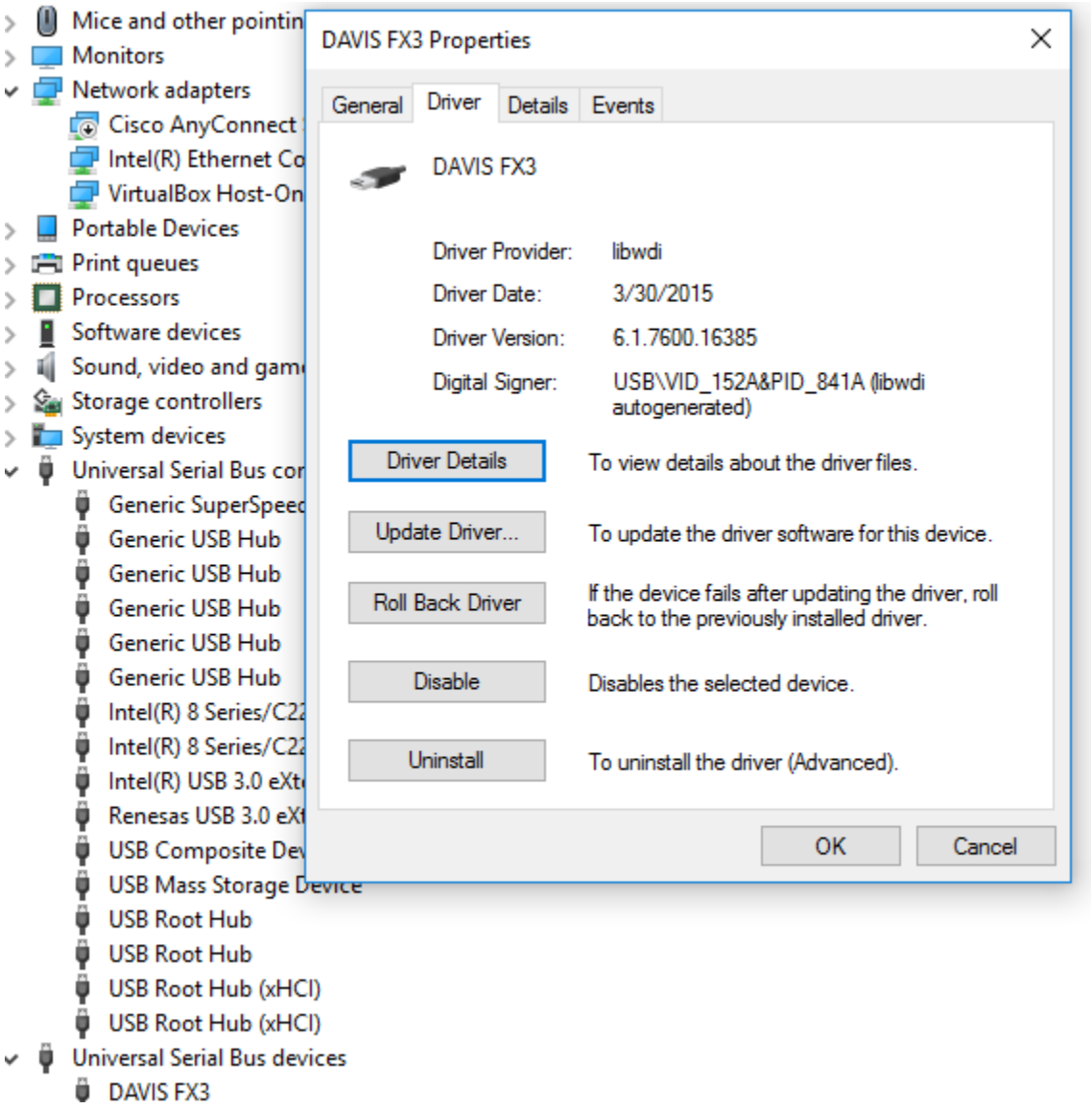
You should now see the correct driver (WinUSB Generic Device) in the Device Manager. The libUSB and libUSBK drivers should NOT be installed; they are intended for development of applications using the libusb-win32 or libusbK APIs.

After driver installation, you should see the driver installed in the Device Manager as shown below:

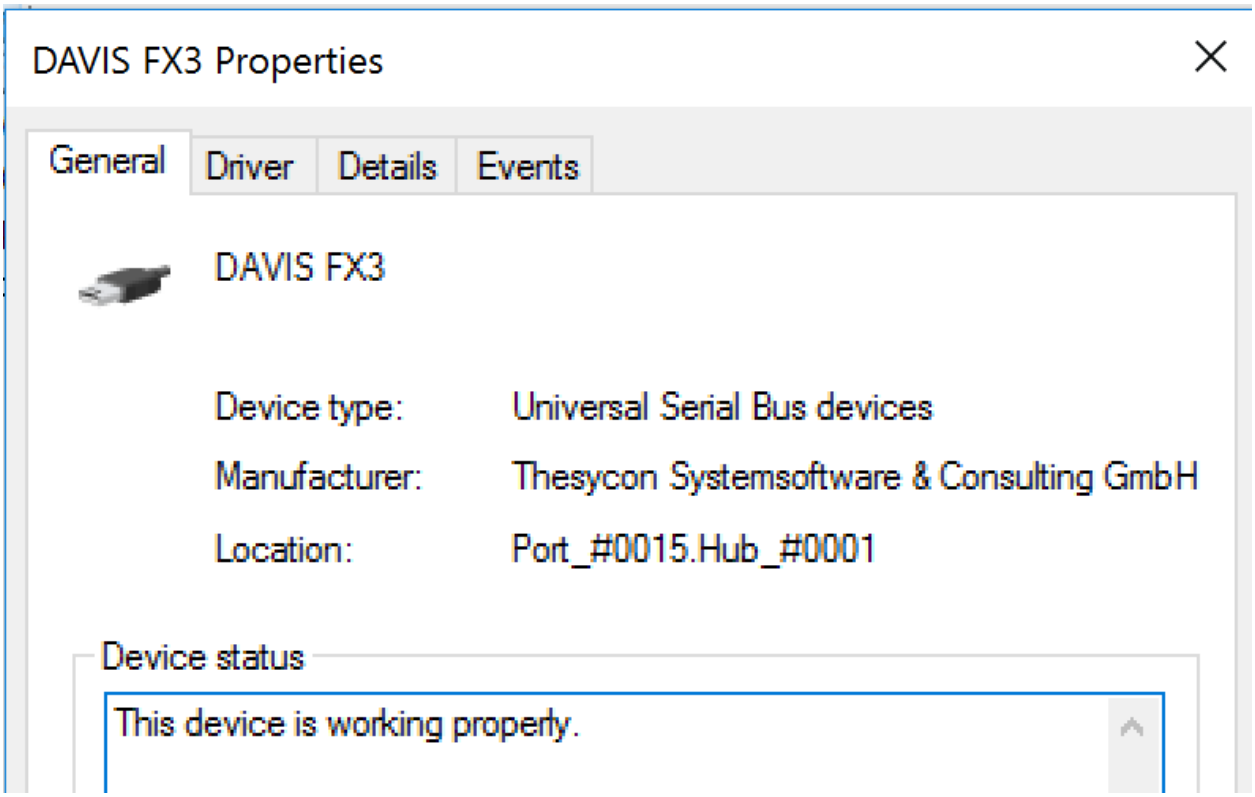
Windows 7 example



Windows 10 example

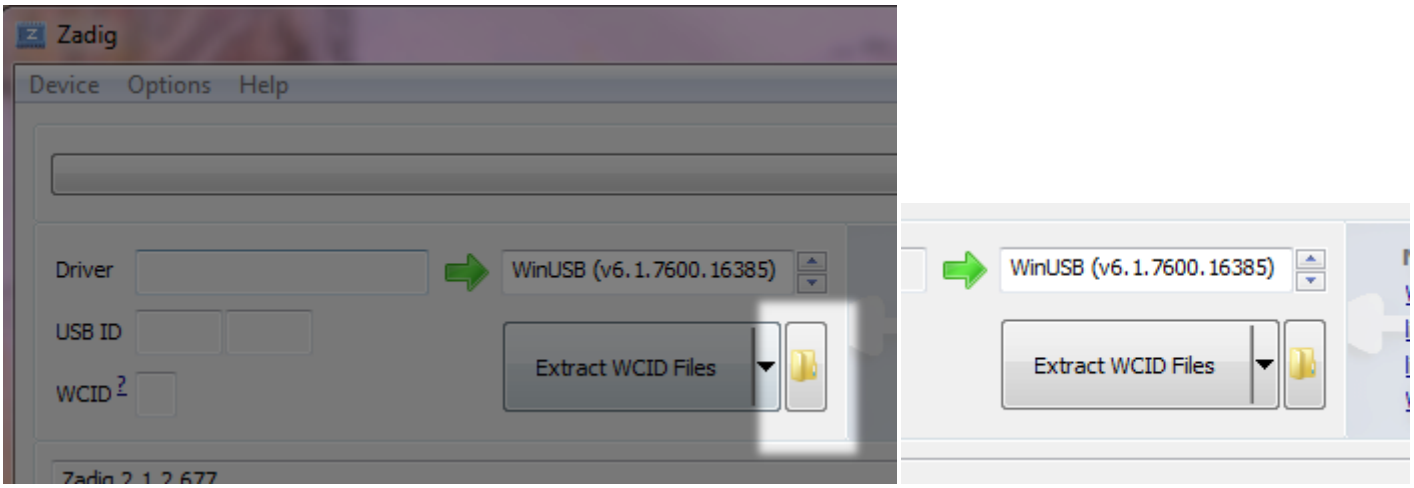


Note that because iniVation uses a vendor/product identification (VID/PID) range purchased from Thesycon, Windows will identify the device's manufacturer as shown below; this is correct.



Troubleshooting Zadig Driver Installation

1. We have seen that in some cases, after taking these steps, you need to restart your computer in order for our software to properly access the devices.
2. We have also seen that Zadig does not list the device even when it is shown as "Unknown Device" in the Device Manager. In this case, you may need to extract the WinUSB driver manually. In Zadig, switch to *Options/Advanced mode* and use the folder icon to select a folder to extract the driver files to. Then use the usual Windows driver installer dialog to search this folder for the driver.

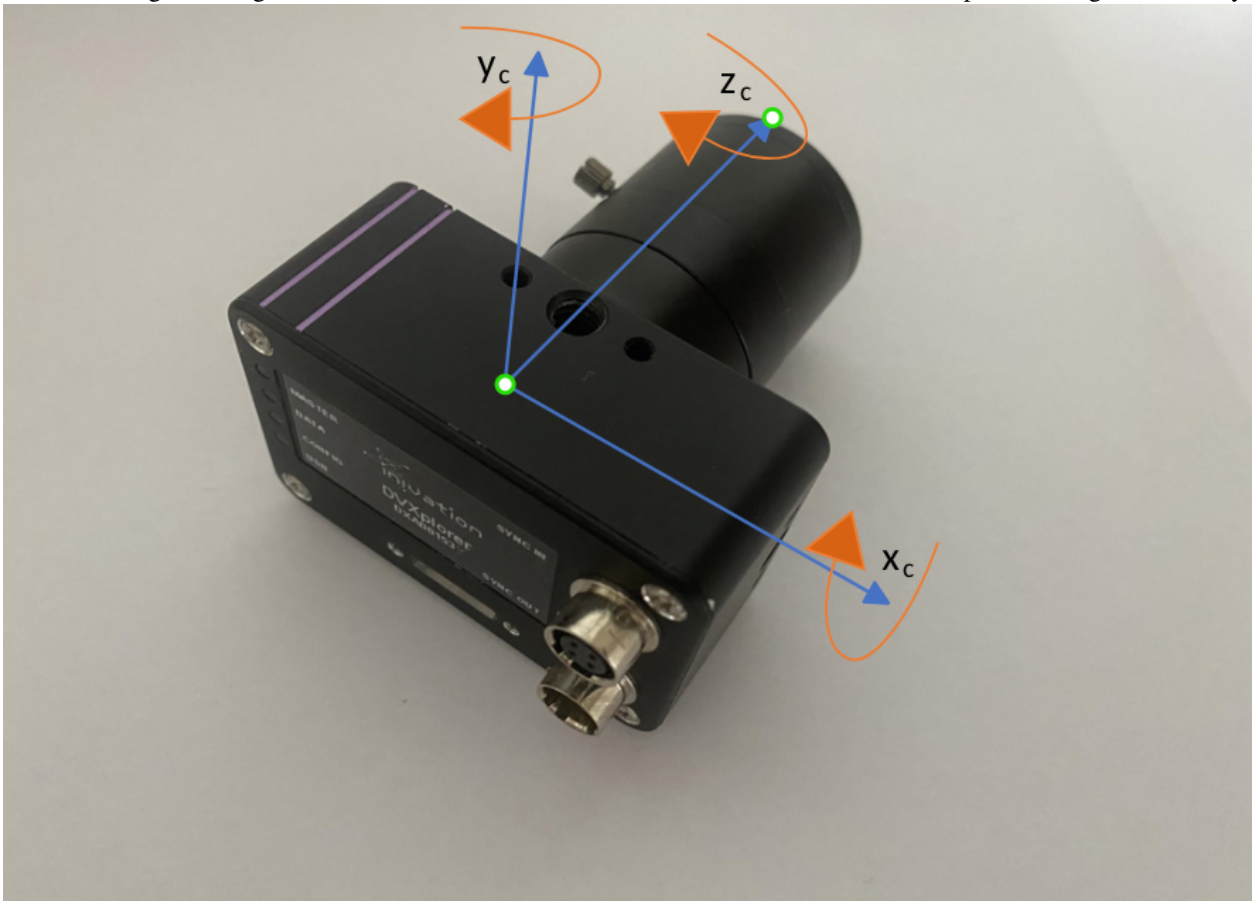


3.4 IMU

The IMU used in our DAVIS346 camera is the TDK InvenSense MPU-6500 (Datasheet[46]). All other cameras use Bosch BMI-160 imu (Datasheet[47]). Before continuing notice that we refer to image sensor as the sensor outputting events, or events and frames in case of DAVIS346.

For several applications related to tracking, IMU can be useful. This sensor provides both angular speed and linear acceleration information. Our cameras output angular speed as (degrees/second) and linear acceleration using gravity acceleration constant as unit (g). Note that to convert acceleration to m/s^2 you need to multiply the accelerometer values by the gravity constant (1g = 9.81m/s^2). In order to be able to combine these data with image-based motion estimation, the relative position and orientation between the image sensor and the IMU is needed.

The following drawing shows the camera axis and the rotation directions for positive angular velocity.



The image sensor uses the standard format: by looking at the camera from the back, X-axis points to the right, Y-axis up and Z-axis points towards the lens. In addition, the IMU axis are aligned with the camera one. The rotation for the gyroscope is counter-clockwise along the increasing axis, for all three axes. Note that, this does not follow the right-hand rule.

3.5 Biasing Dynamic Sensors

This guide explains how to control the on-chip parameters (“biases”) of our devices.

[46] https://docs.inivation.com/_static/imu/mpu6500.pdf
[47] https://docs.inivation.com/_static/imu/BMI160.pdf

3.5.1 What Is A Bias?

Analogue electronic circuits are often parameterised by currents or voltages which are held steady during operation - these currents or voltages are called biases.

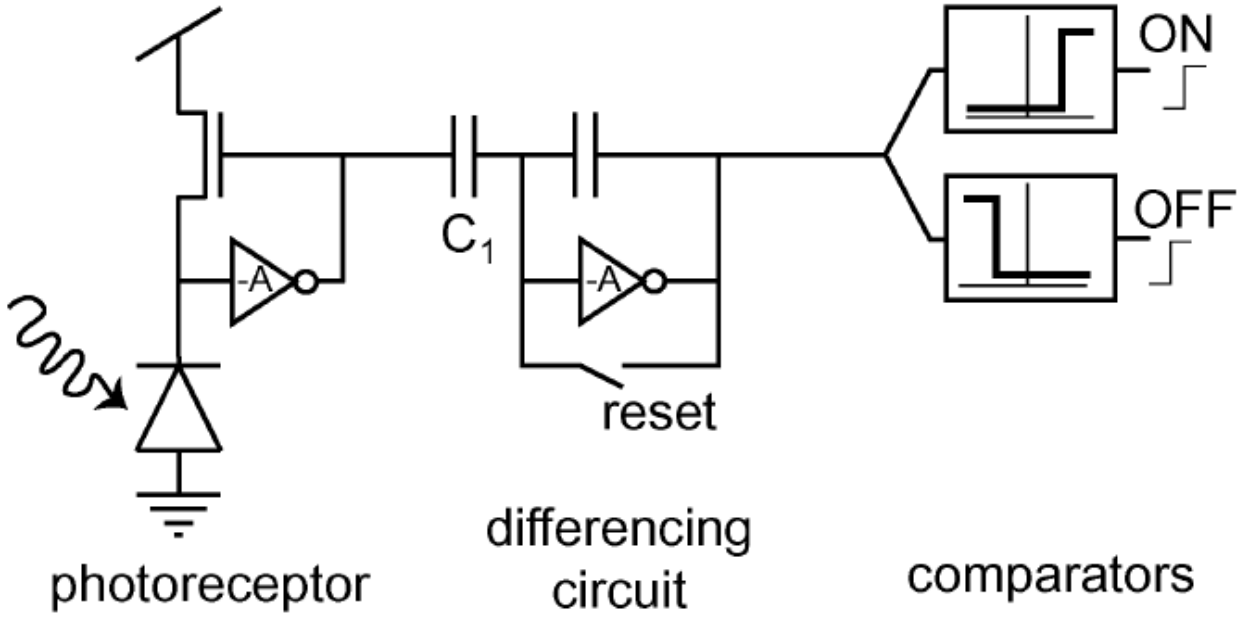
These parameters are used to configure sensor sensitivity and response time when generating events.

An introductory paper on the topic is available here[48].

For even more bias-related information, refer to the Advanced / More Biases-Related Information section.

Understanding The DVS Pixel

A basic understanding of the DVS pixel design is necessary to work effectively with biases; detailed knowledge of analogue electronics is not required. Here is an abstracted diagram of the DVS pixel (taken from Lichtsteiner et al. 2008):



This shows that there are 3 parts of the DVS pixel:

- The first stage is the photoreceptor; there is an amplifier whose job it is to stabilise the voltage across the photodiode and create a voltage signal which is proportional to the log of the light intensity (the "light-related signal").
- The second stage, the "differencing circuit", rejects the DC component of the light-related signal whenever it is reset, so that the resulting signal doesn't carry information about the absolute level of illumination.
- The third "comparators" stage detects changes in the light-related signal and produces digital events (ON means the light got brighter and OFF means it got dimmer).

Pixel Bandwidth VS Chip Bandwidth

It's worth disambiguating these terms. The chip bandwidth is the maximum frequency with which events can be transmitted from the chip, if lots are produced by pixels at the same time so that they are queueing inside the chip to be transmitted. For example, this is about 165 M Events Per Second (MEPS) for the DVXplorer and about 12 MEPS for the DAVIS346.

The term "pixel bandwidth" has sometimes been used in different ways in the DVS literature. It might relate to:

[48] https://doi.org/10.48550/arXiv.2304.04706

- The maximum frequency of an oscillating illumination to which the pixel could reliably produce a series of ON and OFF events.
- The maximum frequency at which a pixel can produce events in response to fast changes in illumination.
- The briefest transient change in illumination that could result in the production of an event.

3.5.2 DVXplorer Biases

On DVXplorer devices (*DVXplorer*, *DVXplorer Lite*, *DVXplorer Micro*), biases are "hidden" and grouped under two simple and explicit parameters to be controlled by the user:

- `contrastThresholdOn and constrastThresholdOff`. Used to control the sensor sensitivity to intensity changes to generate ON and OFF events. The range is from 0 to 17, with the middle value 9 being the default. Smaller values mean a smaller contrast change will be required to generate an event (more events, more sensitivity, more noise), while bigger values do the opposite, requiring a bigger change in contrast before an event is generated (less events, less sensitivity, less noise).
- `ReadoutFPS`. Used to control the number of event frames per second in sensor readout (read more in our *White Paper*). There are many values available, they are named using **Readout mode** and **FPS value**:
  - `CONSTANT`, with this **Readout mode**, the readout time is guaranteed to be constant (at 1/**FPS value**) with no data loss. It is only supported as the following values:
    - `CONSTANT_100`: *10000* µs readout time per frame.
    - `CONSTANT_200`: *5000* µs readout time per frame.
    - `CONSTANT_500`: *2000* µs readout time per frame.
    - `CONSTANT_1000`: *1000* µs readout time per frame.
  - `CONSTANT_LOSSY`, with this **Readout mode**, the readout time is guaranteed to be constant (at 1/**FPS value**) but it might be smaller than the actual time needed for the full readout (max ~900 µs). There might be some data loss depending on the event load. It is only supported as the following values:
    - `CONSTANT_LOSSY_2000`: *500* µs readout time per frame.
    - `CONSTANT_LOSSY_5000`: *200* µs readout time per frame.
    - `CONSTANT_LOSSY_10000`: *100* µs readout time per frame.
  - `VARIABLE`, with this **Readout mode**, the readout time will try to match the set value (at 1/**FPS value**) but it may be increased depending on the event load. It is only supported as the following values:
    - `VARIABLE_2000`: *500* µs minimum readout time per frame.
    - `VARIABLE_5000`: *200* µs minimum readout time per frame.
    - `VARIABLE_10000`: *100* µs minimum readout time per frame.
    - `VARIABLE_15000`: *66* µs minimum readout time per frame.

3.5.3 Davis Biases

On Davis devices (*Davis346*, *Davis346 AER*), the different biases are exposed to the user and have varying purposes. This guide will not tell you what exact biases and values to use. It will rather help you to understand the effects of the biases and guide and constrain your search for the optimal biases for your applications.

Understanding And Computing Biases

On Davis devices, biases are currents that are created by a two-stage coarse-fine circuit[49].

- The first stage (the `coarse` current value) provides a choice of 8 currents (`coarse` value between `0` and `7`) which are logarithmically spaced over more than 6 orders of magnitude, from 25 uA down to 12 pA.
- The second stage divides the `coarse` current value by a digitally programmable value, the `fine` value.
- Using the `coarse` and `fine` biases, the maximum value obtainable is that of the coarse bias, whereas the minimum value achievable is zero.

Each Bias is in total composed of 6 values:

- `Coarse value`, as explained above
- `Fine value`, as explained above
- `Enabled`, whether the bias is enabled or not
- `Sex`, whether the bias is applied to an N- or P-type transistor
- `Type`, whether the bias is arranged "normally" or uses a Cascode.
- `Current Level`, whether the current level is "normal" or "low".

The `Current Level`, `Type`, `Sex` and `Enabled` options are not meant to ever be changed and are not available to users.

All the previously listed values are stored and set to the device through a single 16-bit integer (one for each bias). Its format is the following:

```
UCCC FFFF FFFF LCSE

U: Unused

C: Coarse value

F: Fine value

L: Current Level: 1 = Normal; 0 = Low

C: Type: 1 = Normal; 0 = Cascode

S: Sex: 1 = N-type; 0 = P-type

E: Enabled: 1 = Enabled; 0 = Disabled
```

To compute or convert bias values following the previous scheme, one can use the libcaer functions[50] or the corresponding Excel sheet (Download[51]).

The interested reader is referred to Delbruck et al. 2010[52], where the design of the bias generator used on the DAVIS is described in detail.

[49] https://doi.org/10.1109/ISCAS.2012.6271979
[50] https://gitlab.com/inivation/dv/libcaer/-/blob/master/src/davis.c#L805-840
[51] https://docs.inivation.com/_static/biasing/math.xlsx
[52] https://doi.org/10.1109/ISCAS.2010.5537475

Important Biases

Photoreceptor Bias

Naming: Davis='PrBp', DVS='Pr'

This bias controls the amplifier in the first stage, and limits the speed with which the output of the first stage can respond to changes. An instantaneous change in illumination causes a change in the light-related signal which takes a finite time to readjust. This finite time is highly variable (from microseconds to milliseconds) and it depends on two factors: the level of illumination and this *Pr* bias. The interaction is as follows. With low illumination and a sufficiently high *Pr* bias, the adjustment time is dictated by the light level. With high illumination or a low *Pr* bias, the adjustment time is dictated by the *Pr* bias. This means that you can use the *Pr* bias to ensure that this response time is slow, but in order to guarantee a fast response time you need both a high *Pr* bias and sufficient scene illumination.

Implications for pixel bandwidth and noise: the speed with which a pixel can respond to changes in light (the "bandwidth") is dictated by several factors; the *Pr* bias and the scene illumination are two of these factors. If the pixel bandwidth is high then it will detect faster oscillations of illumination; however it will also respond to higher frequency electronic noise, therefore producing more noise events (especially in low lighting conditions).

Source Follower

Naming: Davis='PrSFBp', DVS='foll'

Between the first and the second stage, not shown in the above diagram, there is a circuit whose job is to pass the signal from the first stage through to the second stage whilst reducing coupling from the second stage back to the first stage (this is a "source follower", a type of amplifier). This bias dictates the speed at which this amplifier works. If this bias is set high, it would allow a high pixel bandwidth for fast detection, and at the same time, introduce increased in-band noise, and hence result in increased noise events. However, if this bias is low then it can limit the bandwidth of the pixel in much the same way as the *Pr* bias can.

Differential

Naming: Davis='DiffBn', DVS='diff'

This bias controls the amplifier in the second stage. Unlike the *Pr* bias which has a complex interaction with illumination level, this bias completely determines the speed at which the second stage adjusts to a change in the light-related signal.

Thresholds For On And Off Events

Naming: Davis='OnBn/OffBn', DVS='diffOn/diffOff'

The size of the change in illumination necessary to produce events is set by varying biases for thresholds. These are independent for increases and decreases in illumination. This is also called setting the "contrast sensitivity" - large thresholds imply a low contrast sensitivity and *vice versa*.

When the pixel is reset, the output of the second stage to the comparators is a value set by the *diff* bias. (Here we talk about a "value" for simplicity; the value is a hypothetical current - one which the comparators try to recreate). Once the light-related signal changes, this value changes. If there is more light, the value gets higher and if there is less light then the value gets lower. The change in this value is proportional to the change in illumination, multiplied by the gain of the amplifier. In both DVS128 and DAVIS, the gain is about 20X, so a doubling of the amount of light means a 20-fold increase in this value.

The *diffOn* bias defines the current level at which the pixel will produce an ON event. This must always be higher than the *diff* bias, and the ratio between the two defines the change in light level necessary to produce an event. For example, if *diffOn* is 10 times as big as *diff*, then a 50% increase in illumination should trigger an event, i.e. ratio / gain = 10 / 20 = 50% relative change expressed as a percentage.

Likewise, the *diffOff* bias defines the current level at which the pixel will produce an OFF event. This must always be **lower** than the *diff* bias, and the percentage change between the two defines the change in light level necessary to produce an event.

Due to mismatch, if you bring either *diffOn* or *diffOff* too close to *diff* then you will see some pixels malfunctioning and at some point the sensor will fail to perform.

Refractory Period

Naming: Davis='RefrBp', DVS='refr'

When a pixel crosses a threshold it signals to peripheral circuitry in first one dimension then the other. This takes a finite amount of time (typically less than 1 us, although when more than one pixel fires at a time, this time can extend due to queueing). Once the pixel receives the knowledge in both dimensions indicating that communication was successful, it resets itself. This reset requires a finite amount of time (partially dictated by the *diff* bias).

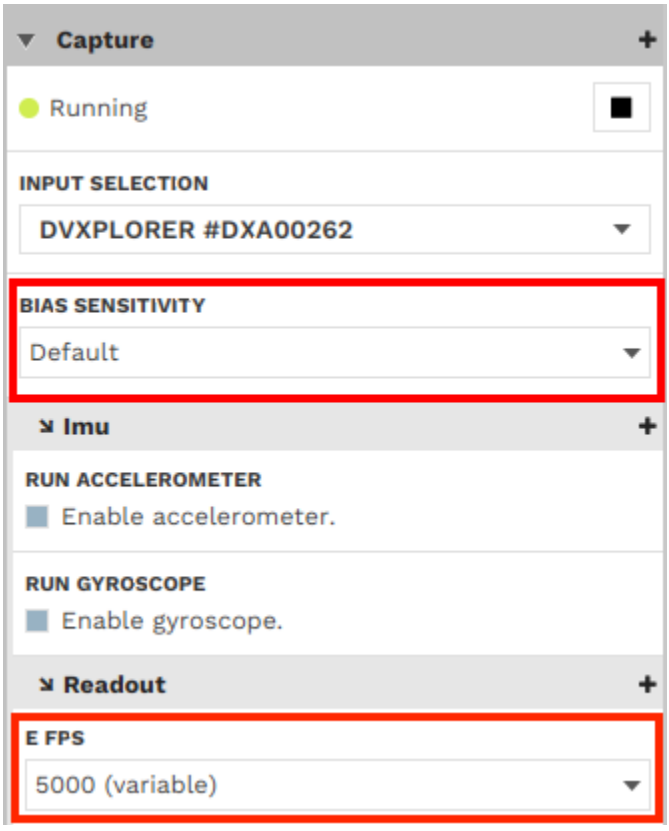
The *refr* bias defines the time period during which the pixel will be reset, before it can again start to detect changes in the light-related signal coming from the first stage. Note that this does not stop the first stage from producing the light-related signal - this happens continuously. Thus, any changes that occur during the time it takes for a pixel to first communicate its event and then reset itself are ignored.

We use the term "Refractory period" because of the analogy between a DVS pixel and a nerve cell; in neurobiology this term is commonly used to describe this phenomenon.

3.5.4 How To Change Biases

In DV

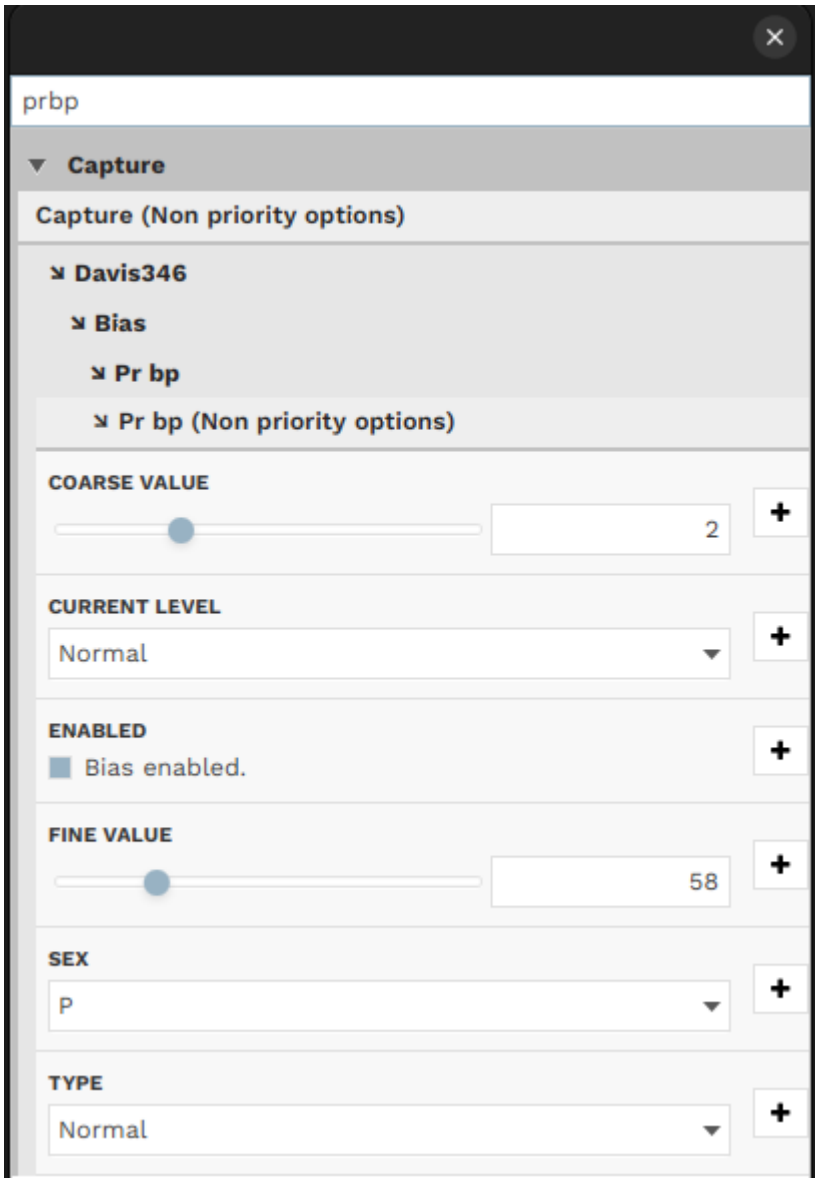
For DVXplorer Devices



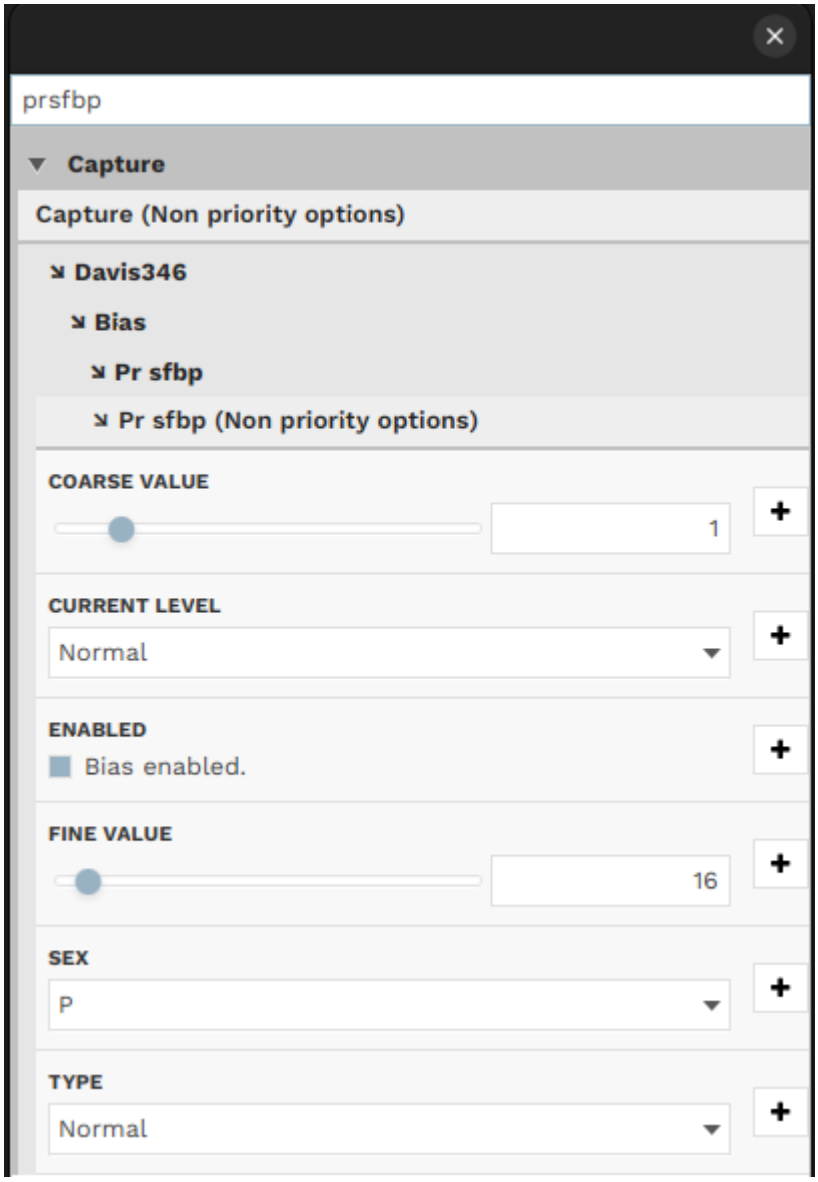
For Davis Devices

To change a Davis bias in DV, access the camera input module *advanced setting*, then type the name of the corresponding bias in the search bar.

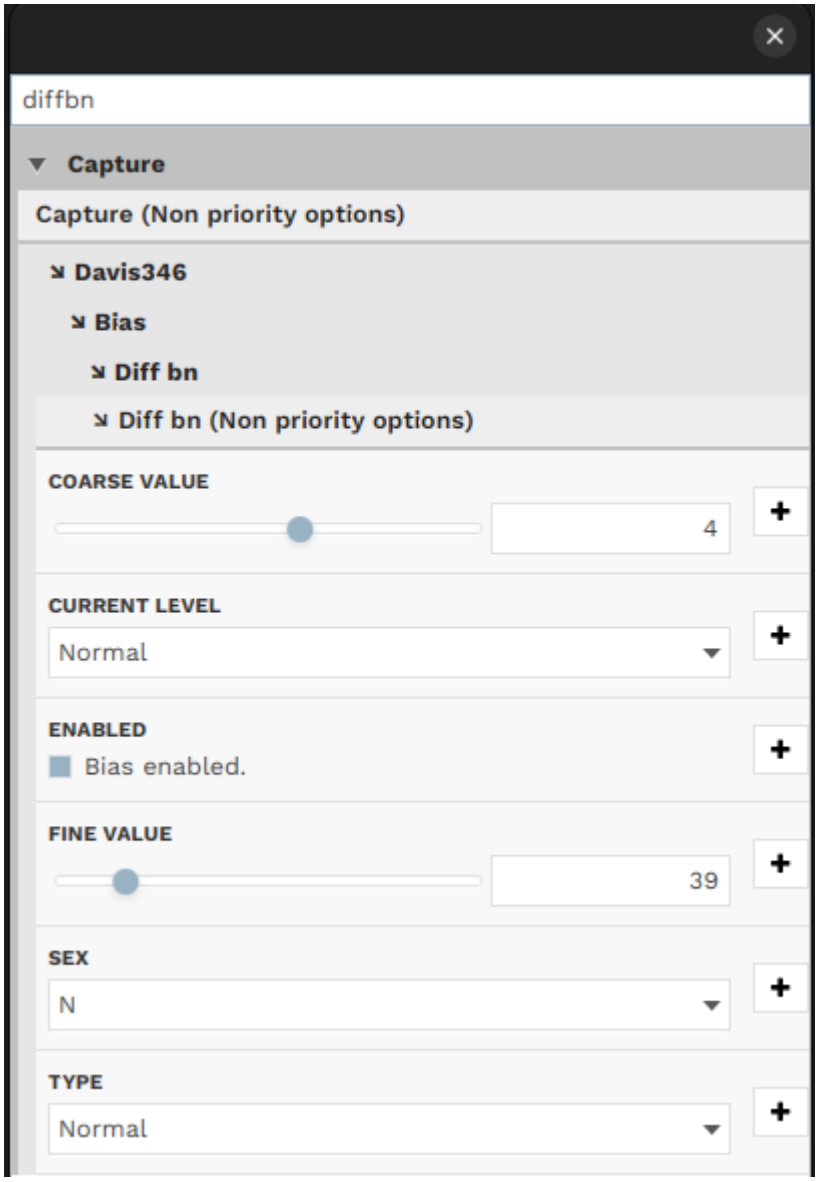
- *Photoreceptor Bias*:



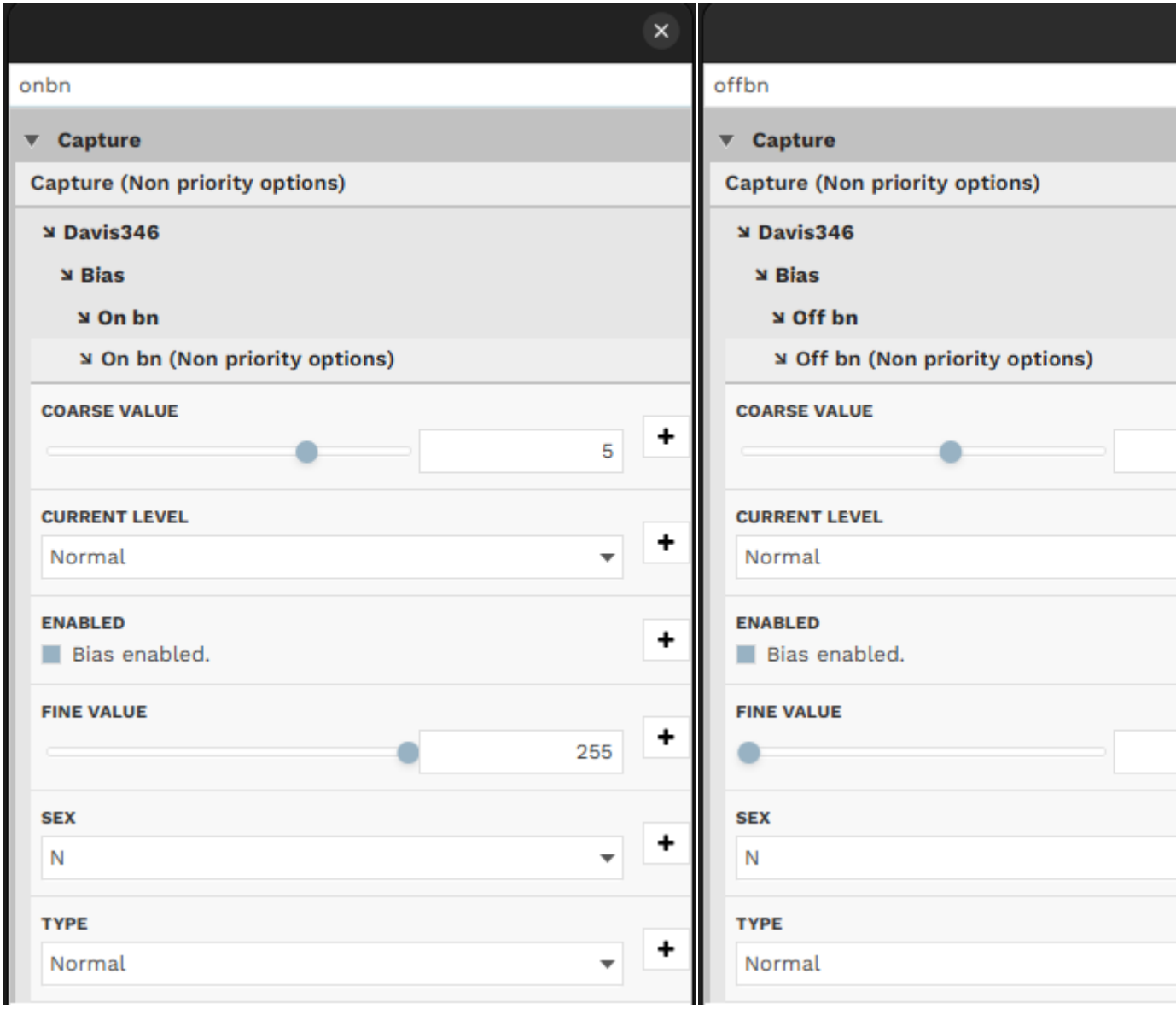
- *Source Follower*:



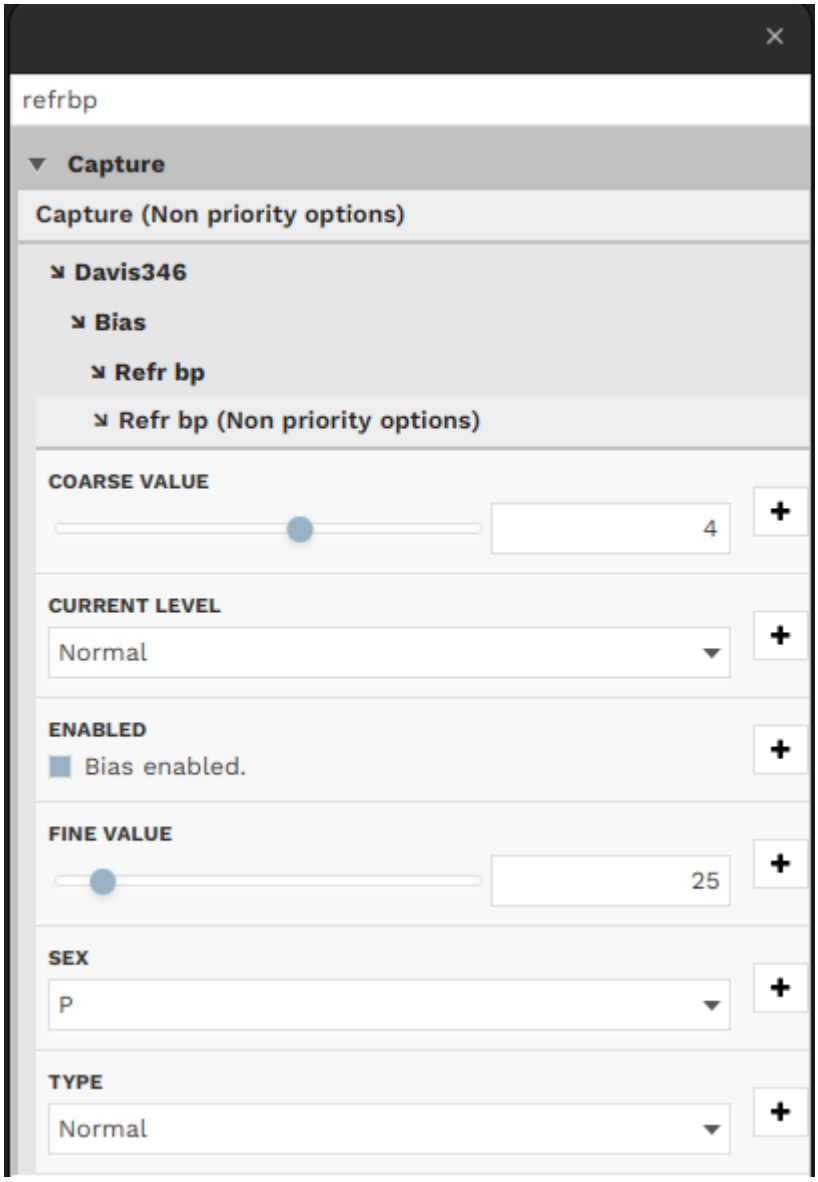
- *Differential*:



- *On/Off Thresholds*:



- *Refractory Period*:



In dv-processing

All instructions on how to change biases in dv-processing can be found here[53].

3.5.5 How Are Biases Generated In Our Sensors?

Our dynamic sensors contain digitally programmable bias generators which can produce currents that can vary over many orders of magnitude (from uA down to fA). These currents then produce voltages which can be distributed across a chip to bias many circuits at once, such as pixels.

Having the bias generator on the chip allows us to eliminate the effects of process, voltage and temperature (PVT) variation between chips. Two different instances of one of our sensors will have very similar behaviours if programmed with the same biases, despite significant differences in the fabrication of devices.

The currents produced by the bias generator vary proportionally to absolute temperature - this is a form of temperature dependence which ideally results in constant transconductance operation across temperature, so that the time constants

[53] https://dv-processing.inivation.com/master/reading_data.html#configuring-camera-options

do not vary with temperature.

Our sensors are digitally programmed on startup from the attached USB microcontroller. They can be reprogrammed dynamically in milliseconds in order to alter their behaviour.

For the interested reader, the bias generator circuitry is explained in:

- Delbruck and Van Schaik 2004[54].
- Delbruck and Van Schaik 2005[55],
- Delbruck et al. 2006[56],
- Delbruck et al. 2010[57],
- Yang et al. 2012[58]

3.5.6 Advanced

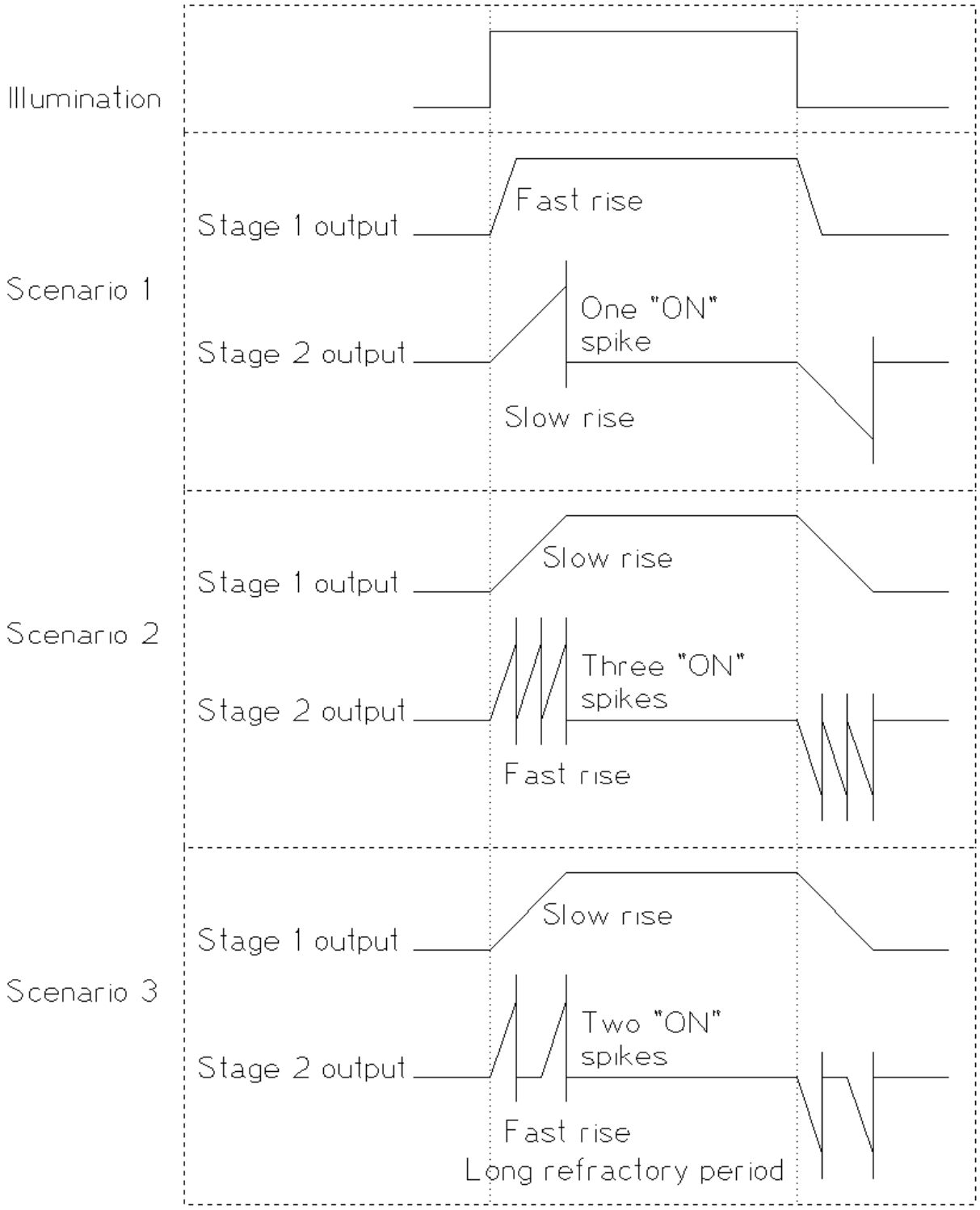
More Biases-Related Information

One Event Or Many, In Response To A Change In Light?

If there is a large change in the amount of illumination, will a pixel produce only one event, or will it produce many events in proportion to the size of the change? Depending on the application, one or the other of these behaviours may be preferable. In fact either of these behaviours are possible, and you have some ability to choose between these qualitatively different modes of operation.

We have seen that the speeds of the first and second stage circuits can be modified independently from each other. Consider the example shown in this diagram:

[54] https://doi.org/10.1109/ISCAS.2004.1328200
[55] https://doi.org/10.1007/s10470-005-1606-1
[56] https://doi.org/10.1109/ISCAS.2006.1693218
[57] https://doi.org/10.1109/ISCAS.2010.5537475
[58] https://doi.org/10.1109/ISCAS.2012.6271979



Let's assume that there is an instantaneous increase in light which is large enough to produce three threshold changes, given the choice of the biases which control the thresholds. There then follows a step back down. This is shown in "Illumination".

In the first scenario, the DVS is biased with a high *Pr* and *PrSF* and a low *diff*. Stage 1 quickly adapts to the new illumination level. Stage 2 takes longer. By the time it has produced its first spike, the stage 1 output has already settled

and there is no more change to detect.

In the second scenario, the DVS is biased with a low *Pr* and *PrSF* and a high *diff*. Stage 1 takes a long time to adapt to the new illumination level. In this time, Stage 2 is able to fire and reset 3 times.

The third scenario is similar to the second, except that the *refr* bias has been set lower, resulting in a longer refractory period. The stage 2 produces its first event quickly in response to the slow rise of stage 1, but then has to wait until its reset completes. In this time it misses some of the rise of the stage 1 output, so that in the end it only produces two spikes in response to the change.

Remember that the production of lots of events across the chip can cause a delay which may have the same effect as the refractory period in the third scenario. The time to spike, the delay in communication and the refractory period all add up to the period of the pixel, from which the pixel bandwidth is derived.

Sources Of Noise

There are three different reasons why you may see noise in the event stream:

- Electronic noise
- Background events
- APS crosstalk

Electronic Noise

The photodiode and each of the transistors all contribute some electronic noise. In the complete absence of light there is still a small current across the photodiode - this is called the dark current; this current has a certain amount of intrinsic noise. As light level increases, the noise in the photocurrent increases, but it does not do so exponentially, with the effect that there is less noise in the light-related signal, which represents the log of the photocurrent. Thus, especially in low-light conditions, and in darker areas of the scene, the pixels may produce spurious events.

From the second stage onwards there is significant amplification of the signal from the first stage (about a factor of 20), so that any noise introduced to the signal is ignorable compared to the contributions from the devices in the first stage.

The power of the electronic noise is distributed across different frequency bands; if the *Pr* or *PrSf* biases are use to limit the bandwidth of the first stage then this reduces the amount of noise that is transduced into spikes (i.e. high-frequency noise is ignored, because the transients are too quick to be detected but rather are smoothed out).

Setting a higher threshold means that only larger deviations in the signal produce events, thus reducing sensitivity to noise at the expense of reducing the contrast sensitivity.

If there is a lot of electronic noise you will see both ON and OFF events coming from pixels. If there is less noise or you have set high thresholds, you may see an occasional ON event from a pixel, quickly followed by an OFF event, or vice versa. This is because the light-related signal is moving around an average value and if there is an occasional large deviation upwards it should be followed by a return towards the average.

Background Events

In well-lit conditions with little electronic noise, there will nevertheless be noise events. These are all of ON type, and from each pixel they arrive with a certain regularity. The rate may be around 1 every 10 seconds or so, depending also on the ON threshold. This is due to a non-ideality in the second stage circuit in which the light-related signal drifts in a given direction in the absence of actual change. The reader seeking a detailed explanation is referred to Lichtsteiner et al. 2008[59] section III-C. The strength of this drift is also strongly dependent on the amount of illumination - more illumination means more frequent events. The background event rate for pixels directed towards the sun can be 100s of Hz.

[59] https://doi.org/10.1109/JSSC.2007.914337

There is no way to eliminate these events, although their frequency can be reduced by setting high thresholds. It's quite common to eliminate both electronic noise and background events in software by applying the BackgroundActivity filter, as described in the jAER code repo[60].

The drift in the second stage that leads to background events also creates a bias towards ON events. The more infrequently events are produced on average, the more noticeable this bias is. For some applications it may be useful to compensate for this with a higher ON threshold. If event rates are high on average then background events do not occur, because pixels do not have enough time to drift between spikes related to changes in scene illumination.

APS Crosstalk

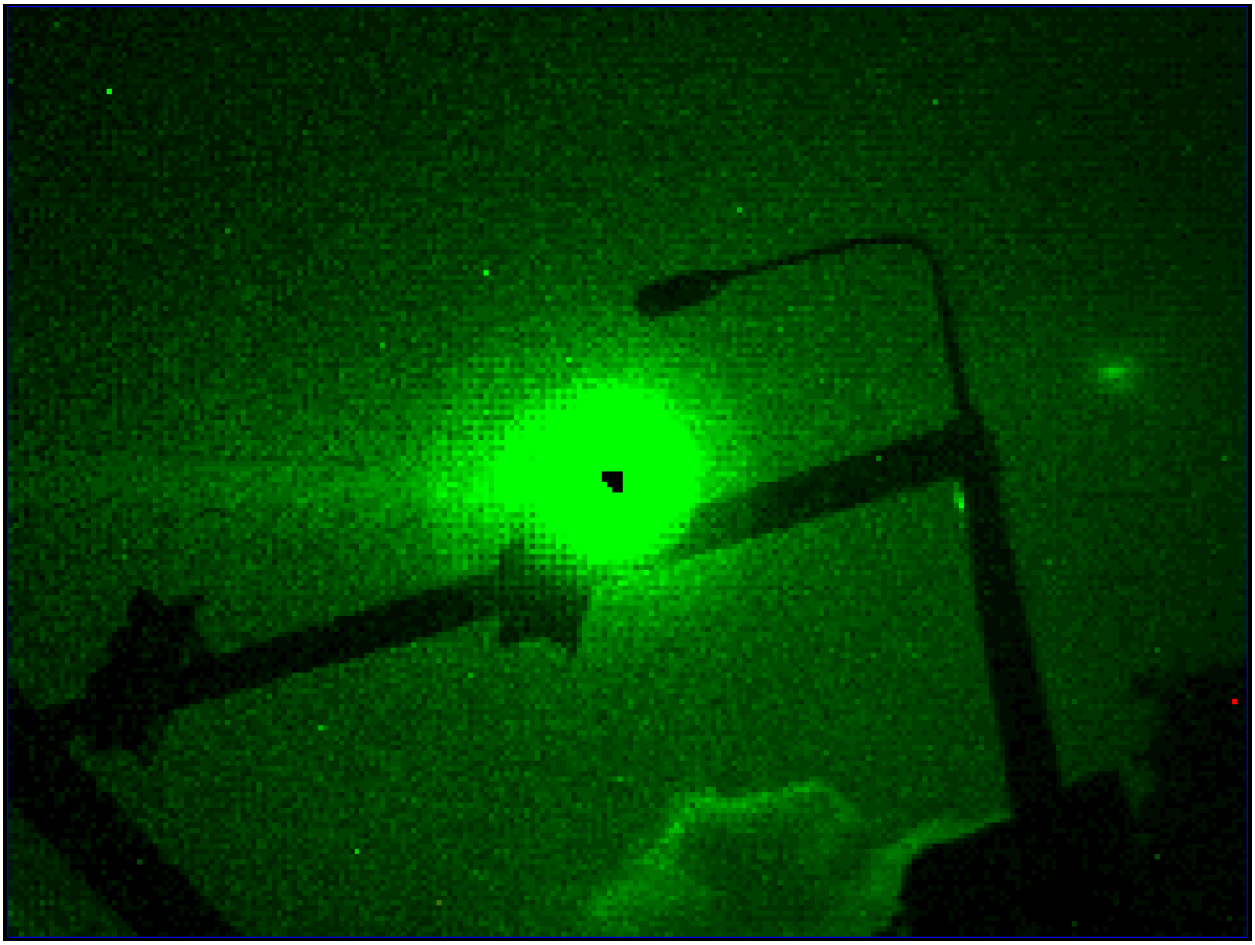
When a global APS exposure is performed, there can be a burst of excessive events correlated with this. These events have a typically noisy characteristic, although some correlation to expected activity can also be seen, i.e. pixels which were about to spike anyway may be induced to spike by the exposure.

This is caused by an undesirable coupling between certain nodes within the pixel. Reduction or elimination of this may be expected in future designs. Reducing contrast sensitivity can help to reduce this problem.

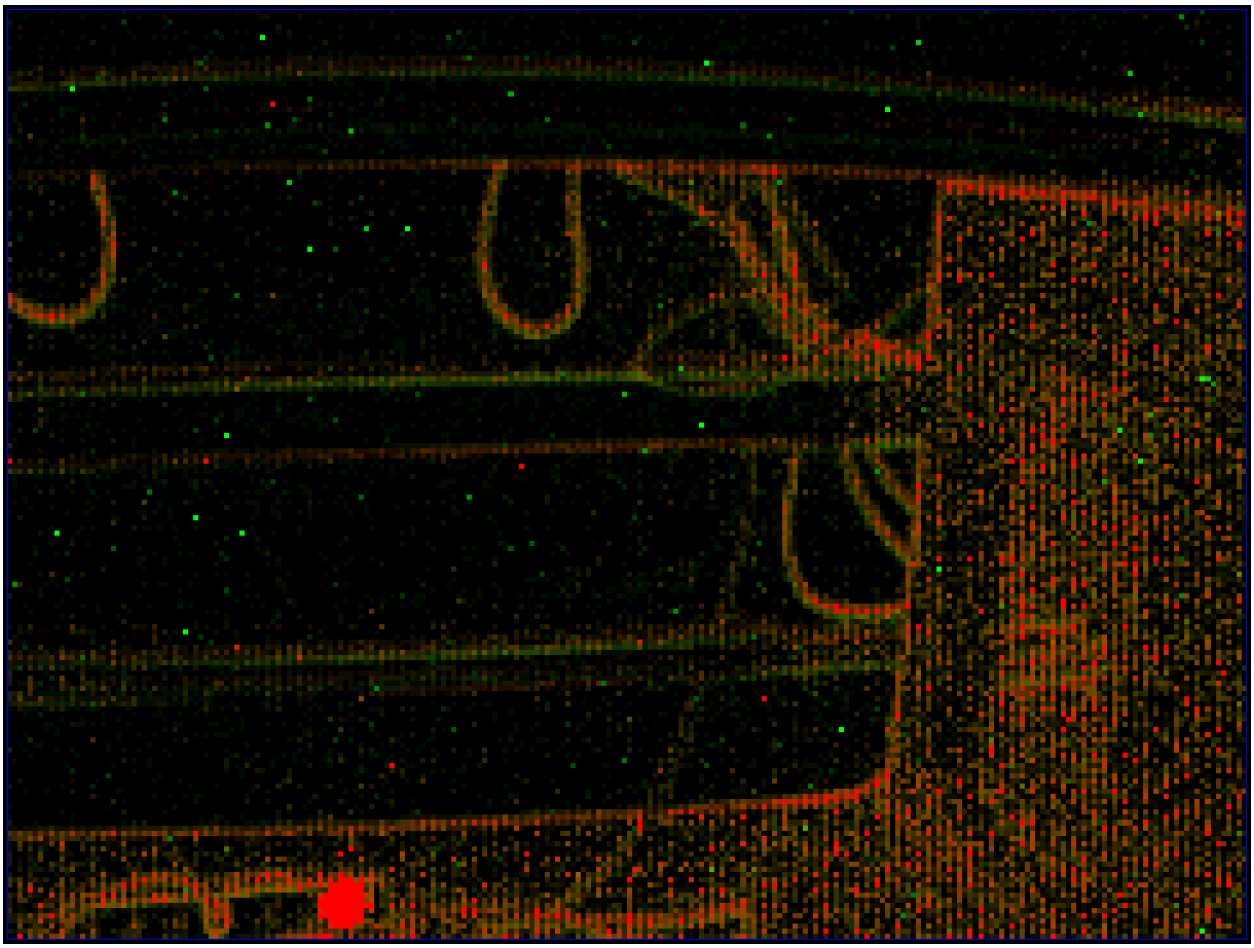
Accumulation Of Noise Can Indicate Light Intensity

Even without APS frames, it's nevertheless possible to obtain some information about intensity. We have seen that in bright lighting, noise is dominated by regular ON events, which are more frequent from brighter regions. The following image was created by accumulating all the ON events over several seconds whilst observing the sun at a traffic light. The bloom around the sun is apparent:

[60] https://github.com/SensorsINI/jaer



On the other hand, in dark conditions, noise is dominated by irregular ON and OFF events, which are more frequent from darker regions. The following image was created by accumulating all the ON and OFF events over several seconds whilst observing a rats nest of cables under a desk. In this case, the cables are black against a white wall, and produce more events. The additional events from a light to the bottom of the image are due to the light being driven by an oscillating current:



Implications For Power Consumption

The power consumption of the DVS / DAVIS chip is usually dwarfed by the power consumption of the supporting circuitry (CPLD and USB protocol chips) - a typical power consumption for a DVS chip is order 10 mW, whereas the whole device may consume >> 100 mW. However, as embedded applications are explored, the need may arise to limit power consumption of the sensor itself.

Where biases are used in the pixel to create currents, there is one copy of the current for each pixel. Most of these biases do not recreate the same current in the pixel except in specific conditions. For example, the *diffOn* bias is typically set to quite a high value; however the full current only flows briefly as the pixel approaches the crossing of the ON threshold.

Some currents, however, run continuously. An example is *foll* / *PrSf*. If this is set to the maximum value - 25 uA - then across the chip there are 240 * 180 = 43200 copies of a 25 uA current being consumed, i.e. approx 1A, at 1.8V = 1.8W! In some cases it is possible to overload the USB power supply by setting biases inappropriately, leading to the device being rejected by the host computer.

Full characterisation of chip performance vs power consumption is work in progress - we would welcome collaborators in this respect.

Extra Biases On DAVIS Chips

Source code for all Davis parameters[61]

[61] https://gitlab.com/inivation/dv/libcaer/-/blob/master/include/libcaer/devices/davis.h

Note: "APS" stands for "Active-Pixel Sensor". This is the term for standard CMOS image sensors, in which a transistor in the pixel actively transmits a value related to the light intensity read during an exposure. "DAVIS", in fact stands for "Dynamic and Active-pixel Vision Sensor" - in other words, it is a fusion of DVS and APS circuitry, capable of both producing dynamic events and taking pictures using the same photodiodes. The APS functionality is under control from digital logic external to the chip, including the sequence of exposing and reading out columns and the length of exposures. The following sections only concern the behaviour of the APS functionality that is controlled by the bias generator.

Separation Between APS and DVS

Naming: 'ApsCasEpc'

This controls a cascode which separates the APS and DVS parts of the pixel circuit, providing a stable voltage to the photoreceptor irrespective of whether an exposure is being performed or not. There are a wide range of values over which this should make no difference to the performance of the sensor.

APS Readout

Naming: 'ApsROSFBn'

This biases the source follower that performs column-parallel APS readout at the edge of the chip. This needs to be quite high to give short settling times for exposures; providing it is high, the exact value should make little difference to the performance of the sensor.

APS Overflow Level

Naming: 'apsOverflowLevel'

This sets the lower limit on the range of analogue values that can result from an exposure. However, it has relatively little influence on the range and therefore on the performance of the APS. It does however, need to be quite high to avoid choking the photoreceptor circuit in the case of strong illumination and long exposures.

Differential Cascodes

Naming: 'DiffCasBnc'

This controls cascodes in the differential comparator stages of the DVS circuit. The purpose of the cascodes is to improve the digital swing of the signals that represent ON and OFF events. This bias should be set higher than *diffOn*. There is a wide range of values over which this bias will make no difference to performance.

Extra "Overclocking" Biases

The following biases adjust the speed of the digital AER circuitry; you should not normally need to adjust these. Users who need to turn their chip bandwidth up to 11[62] may experiment with these (the DVS equivalent of overclocking a PC) - we recommend seeking more advice from us in this case.

Passive Load for Digital AER Circuitry

Naming: Davis='PixInvBn', DVS='req'

The actual uses of this bias in the pixel design varies from DVS128 to DAVIS, but the essential role is the same - there are some circuits in the digital AER circuitry of the pixel (passive load inverters) that require a bias for operation. The current dictates the speed at which these circuits work.

[62] https://en.wikipedia.org/wiki/Up_to_eleven

Common Resets for Requests

Naming: Davis='AEPuYBp/AEPuXBp/AEPdBn', DVS='PuY/PuX/reqPd'

Individual elements from the arbiters pull up (or down in some cases) on shared request lines which go to the chip-level logic for sending off chip. There is then a chip-level transistor for each of these lines which pulls the line back down in the absence of a pull up, thus resetting the circuit for the next cycle. These biases control the strengths of these various pull-down transistors, and thus have some effect on the speed at which a cycle of AER transmission completes.

Timeout After Row Event

Naming: Davis='LcolTimeoutBn' (DAVIS Only)

Pixels may see a change in light and signal their row address, but then the change in light may revert before they have a chance to signal their column address. This leads to a row event which is not followed by any column event (The interested reader is referred to Rapha Berner's PhD thesis[63] chapter 7). This bias dictates the amount of time between a row event being sent and the X-arbiter giving up waiting for a column request from a pixel. If this bias is too low, row-only events will take a significant amount of time; if this bias is too high, it's possible that column events might not be sent.

Biases on DVS128

First Stage Amplifier Cascode

Naming: DVS='cas'

The first stage amplifier contains a cascode transistor whose role is to reduce "Miller capacitance" thus improving the speed and stability of the circuit. There should be a wide range of values over which it makes no difference to the performance, providing that it is significantly higher than the *Pr* bias.

Injected Ground

Naming: DVS='injGnd'

This is used for pixel reset to avoid a potential problem (charge pumping leading to spurious resets - the interested reader is directed to Lichtsteiner et al. 2008[64]). The higher this is, the faster a pixel will respond to the acknowledge in order to reset itself. Reducing this bias can cause the chip to output spikes very slowly or stop altogether; above a certain level, this bias will make no difference to performance.

3.6 External Input Triggering and Clock Synchronization

Several iniVation cameras have extra input and output ports to support special multi-camera setups, let's start by defining the supported features and what they do in this context:

3.6.1 External Input Triggering

This feature allows a limited form of interaction with third-party cameras and systems, using the *INPUT_SIGNAL* and *OUTPUT_SIGNAL* ports.

The *INPUT_SIGNAL* port can be configured to detect rising edges, falling edges and pulses originating from another camera or electronic system, and when it detects such an occurence, a special data point is inserted into the data stream coming from the iniVation camera, with a very precise timestamp, allowing data from the camera to be put in relation with events happening externally.

[63] http://e-collection.library.ethz.ch/eserv/eth:5044/eth-5044-02.pdf
[64] https://doi.org/10.1109/JSSC.2007.914337

This feature can detect signals up to a theoretical maximum of 1 MHz. For square wave inputs, we recommend just detecting the rising edges and not going above 100 KHz frequency for reliable results. No debouncing or signal cleanup is attempted.

Best case would be to just send a pulse when the external camera captures a frame, so you can correlate the two. A lot of machine vision cameras support that kind of trigger, for example to flash external LED lights.

This does not change how the iniVation camera itself operates, so it isn't possible to start/stop recordings, configure the camera or trigger frame captures on DAVIS using this feature.

The *OUTPUT_SIGNAL* port simply relays the *INPUT_SIGNAL* port by default, allowing daisy-chaining of this detection feature. Some iniVation cameras also support generating a PWM-like signal on this port as a configurable option. The signal output is not directly related to camera operation; it does not signal changes to the camera state such as starting/stopping a recording or data acquisition. The table below lists all cameras supporting this extra feature.

All of this has to be configured and enabled using our software, the options are usually called 'External Input Detector' for the detection part on the *INPUT_SIGNAL* port and 'External Input Generator' for the optional PWM signal generation on the *OUTPUT_SIGNAL* port.

3.6.2 Clock Synchronization

The clock synchronization feature allows multiple connected cameras to keep their microsecond-precision timestamps synchronized with each other using the *INPUT_CLOCK* and *OUTPUT_CLOCK* ports.

One of the main features of iniVation cameras and their event output is its precise timing, so when multiple cameras are used in parallel to record the same scene (*stereo* configurations), it is important that this timestamp keeps the same value on all cameras. This mechanism is only recommended to keep time synchronized between iniVation cameras and is not intended for third party usage.

Cameras are connected in a daisy-chain, and as soon as a physical connection is established with a cable, they automatically synchronize (details on synchronization protocol *here*). The first camera in the series, the one without a signal on *INPUT_CLOCK*, is called the *Master* camera, and will have its *Master LED* turned on. The other cameras will have the LED turned off. Synchronization status can also be checked in our software, where the option **'deviceIsMaster'** normally informs the user about a camera's status.

Once all cameras are hardware synchronized, you simply have to reset the timestamp of the *Master* camera, which will then reset all the downstream devices too, ensuring that all of them have the same start point in time. This is done via software, either manually by selecting the **'Reset Timestamps'** option or by using *dv-processing's Stereo classes* which do this automatically.

3.6.3 Hardware

The following table contains a quick overview of which iniVation camera models support external input triggering and clock synchronization:

Model	Supports clock synchronization and external input triggering	Supports PWM signal generation (OUTPUT_SIGNAL)	Used connector	Supported voltage
DVS128	YES (old protocol)	NO	2.54mm pins	3.3V
DVS240	YES	YES	HiRose HR10A-7R-4P / HR10A-7R-4S	3.3V-5V
DAVIS240	YES	NO	3.5mm audio jack	5V
DAVIS346	YES	YES	HiRose HR10A-7R-4P / HR10A-7R-4S	3.3V-5V
DVXplorer	YES	YES	HiRose HR10A-7R-4P / HR10A-7R-4S	3.3V-5V
DVXplorer Lite	YES	YES	HiRose HR10A-7R-4P / HR10A-7R-4S	3.3V-5V
DVXplorer Mini	NO	NO	-	-
eDVS	NO	NO	-	-

DVS240, DAVIS346, DVXplorer, DVXplorer Lite

The newest camera models have two connectors on the backplate; the OUTPUT SYNC connector is a HiRose HR10A-7R-4P (male) and the INPUT SYNC connector is a HiRose HR10A-7R-4S (female).

Cables should use the matching connectors HR10A-7P-4S (female) and HR10A-7P-4P (male).

The following supplier can build such cables to order: Raymo[65].

The pinout is shown in the following image:

[65] https://www.alibaba.com/product-detail/HR10A-male-and-female-4P-cable_60631952696.html?spm=a2700.7724838.2017115.66.70e148b1A0GTru



Input signals can be 3.3V or 5V, depending on the VDD_IN supplied externally. **GND_IN and VDD_IN must always be supplied.**

Output signals are always 5V, as is VDD_OUT in relation to GND_OUT. If you chain cameras together for synchronization, the clock and VDD will be 5V, for example.

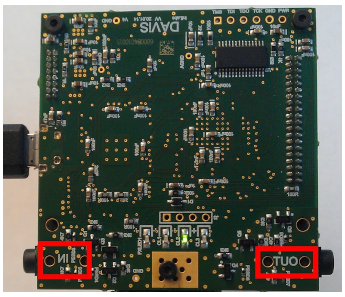
Please note that all the pins in the INPUT SYNC connector are isolated from the OUTPUT SYNC connector. To keep full electrical isolation between different cameras, the cable should not be shielded, or if it is, the shield should not connect one end of the cable to the other.

If the CLOCK_IN signal is not being used, we strongly recommend shorting it to VDD_IN, to provide a stable known value. Failure to do so often leads to coupling on the CLOCK_IN line, for example from a signal on the SIGNAL_IN line, which then leads to spurious timestamp resets on the camera, which can strongly affects its performance.

For the purpose of clock synchronization, as soon as the cameras are connected they will automatically synchronize time and figure out their arrangement; the first camera's *Master LED* (see description on backplate) will remain lit, while the other camera's will turn off.

DAVIS240

Two 3.5mm audio jacks, labeled IN and OUT (printed backwards on the PCBs), are used by these cameras to enable synchronization features.



Any 3.5mm audio cable can be used to connect camers to each-other, and can be found at any AV store or suppliers such as digikey[66].



To access the clock synchronization and external input triggering features separately, or to more easily integrate with third party systems, audio connectors that split the jack into two RCAs can be used:

[66] https://www.digikey.com/



Each audio jack has the following pinout:



- External first circle from the cable, "sleeve": GND (Ground)
- External second circle: SIGNAL
- Tip: CLOCK

The voltage for SIGNAL and CLOCK, both on the IN and OUT sides, is 5V.

For the purpose of clock synchronization, as soon as the cameras are connected they will automatically synchronize time and figure out their arrangement; the first camera's *Master LED* (LED1) will remain lit, while the other camera's will turn off.

DVS128

This camera only exposes a few simple 2.54mm pins: IN, OUT and GND.



You can connect these using simple copper cables or jumper wire.



The signal on the IN pin has to be 3.3V. DO NOT connect this with devices providing a 5V signal, such as the DAVIS240!

The OUT pin will always produce a 3.3V signal.

In the old synchronization protocol, a device must be told via software whether it is a master, which produces the synchronization signal, or whether it is a slave, which receives it. Further, the DVS128 has only one pin each for input and output, leading to a more limited set of modes of operation:

- **camera configured as Master (sender) via software:** the IN pin can be used for external input triggering by detecting rising edges only, the OUT pin will generate the clock synchronization signal.

- **camera configured as Slave (receiver) via software:** the IN pin can only receive the clock synchronization signal from another camera, the OUT pin will repeat the clock synchronization signal.

To configure the camera in these two modes, please first make sure that it is running the latest firmware version 14. By default a DVS128 will be in *Master* mode. The *Master* camera keeps its bottom LED on, while the others have it turn off.

When using dv-processing, you will find a `setMaster()` function in the `dv::io::camera::DVS128` class, respectively in DV you'll find a '*Set Master*' checkbox in the module configuration.

In libcaer (deprecated), you can use the following C++ code to switch modes:

```
deviceHandle.configSet(DVS128_CONFIG_DVS, DVS128_CONFIG_DVS_TS_MASTER, true/false);
```

3.6.4 Software

DV

To view the clock synchronization status for a supported device in ***DV***, open the extended camera module configuration (plus sign in right-hand menu) and search for the '*Device is Master*' checkbox, as well as the '*Sync Status*' field.

To properly synchronize multiple cameras, please execute the following procedure:

1. Make sure all cameras are connected and hardware synchronized, check their LEDs. Only the first **Master LED** should be turned on.
2. Add all required camera modules to DV.
3. Start all camera modules.
4. On the Master clock camera module, open its configuration and edit the 'Sync With' field by adding the module names of all secondary cameras, comma separated (for example: "dvx1,dvx2,davis1").
5. On the Master clock camera module, open its configuration and click on the 'Sync Time' button. You will have to repeat at least this step manually every time you restart DV, even if loading from a saved project.
6. Done! A series of messages should be printed in the log console, one for each camera, all with the same real-time offset. All cameras are now hardware synchronized and use the same real-time offset, all timestamps are directly comparable.

The "triggers" output of the camera module in the DV software will contain the external triggers data, each recorded trigger event will contain a timestamp and a type set to either **EXTERNAL_SIGNAL_RISING_EDGE** or **EXTERNAL_SIGNAL_FALLING_EDGE** depending on detector configuration, as it can react to raising edges and/or falling edges.

To enable the external signal detector in our software, open the detailed configuration of the camera module (plus sign in right-hand menu) and search for 'detector', you'll then be shown the options for it. Turn on rising or falling edge detection, and then press the '*Detector Run*' button. You can then connect the "triggers" output to the file output module ("Record" project) to write the data out to a file, together with the events, and then process them offline.

dv-processing

Open each camera[67] normally, using their device classes (*dv::io::camera::DAVIS*, *dv::io::camera::DVXplorer*, …) or using the function '*dv::io::camera::openSync(serialNumber)*'. Member functions '*isMaster()*' and '*isSynchronized()*' of the device classes allow checking of current camera clock synchronization status.

To synchronize the cameras, pass a reference (or pointer) of each camera to the function '*dv::io::camera::synchronizeAnyTwo(first, second)*'. In case of three or more cameras, use the function '*dv::io::camera::synchronizeAny(…)*'.

[67] https://dv-processing.inivation.com/master/reading_data.html#from-a-camera

C++ example code is available in samples/io/stereo-capture/stereo-capture.cpp[68]. A Python version is available in python/samples/stereo_camera_capture.py[69].

libcaer (deprecated)

To check the clock synchronization status for a device using *libcaer*, use the following C++ code:

```
bool deviceIsMaster = deviceHandle.infoGet().deviceIsMaster;
```

Once you have confirmed correct synchronization between all cameras (only one *Master* camera should be present), you must reset the timestamp on the *Master* camera only so that they all start from a common time point. You can use the following C++ code to reset timestamps:

```
DVS128:
masterDeviceHandle.configSet(DVS128_CONFIG_DVS, DVS128_CONFIG_DVS_TIMESTAMP_RESET,␣
↪true);

DAVIS:
masterDeviceHandle.configSet(DAVIS_CONFIG_MUX, DAVIS_CONFIG_MUX_TIMESTAMP_RESET,␣
↪true);

DVXplorer:
masterDeviceHandle.configSet(DVX_MUX, DVX_MUX_TIMESTAMP_RESET, true);
```

Resetting the timestamp will generate a data packet of type `caer_special_event_packet` with one event of type `caer_special_event`, containing a timestamp and the type set to **TIMESTAMP_RESET**.

Running the external signal detector will result in data packets of type `caer_special_event_packet` with events of type `caer_special_event`, containing a timestamp and the type of the detected signal: **EXTERNAL_INPUT_RISING_EDGE** or **EXTERNAL_INPUT_FALLING_EDGE**.

The C++ code below will enable the detector and configure it to report rising edges in the signal:

```
DAVIS:
deviceHandle.configSet(DAVIS_CONFIG_EXTINPUT, DAVIS_CONFIG_EXTINPUT_DETECT_RISING_
↪EDGES, true);
deviceHandle.configSet(DAVIS_CONFIG_EXTINPUT, DAVIS_CONFIG_EXTINPUT_RUN_DETECTOR,␣
↪true);

DVXplorer:
deviceHandle.configSet(DVX_EXTINPUT, DVX_EXTINPUT_DETECT_RISING_EDGES, true);
deviceHandle.configSet(DVX_EXTINPUT, DVX_EXTINPUT_RUN_DETECTOR, true);
```

3.6.5 Appendix

Clock Synchronization Protocol Details

In protocol 1 (DVS128 only), a device must be told whether it is a master, which produces synchronization pulses, or whether it is a slave, which receives synchronization pulses.

In protocol 2 (all other cameras), a device assumes that it is a master, unless it is receiving synchronization pulses, in which case it assumes it is a slave automatically. This simplifies setup.

[68] https://gitlab.com/inivation/dv/dv-processing/-/blob/master/samples/io/stereo-capture/stereo-capture.cpp
[69] https://gitlab.com/inivation/dv/dv-processing/-/blob/master/python/samples/stereo_camera_capture.py

In both protocols a 10 KHz clock is used to advance the timestamps. In slaves, timestamps are allowed to advance on the falling edge of the clock. If the falling edge is delayed by a small period - up to 16 µs for protocol 1, and up to 10 µs for protocol 2 - then any events in that waiting period continue to take the same timestamp until the falling edge is detected.

After the small period, it is assumed that a falling pulse represents a reset pulse. The reset pulse is supposed to last 200 µs.

In protocol 1, slave devices pause their clock indefinitely until the next falling edge, at which point they reset their clock and then continue to increment it. In protocol 2, if the falling edge is not received within 210 µs from the previous falling edge, then the device assumes that it has become a master itself, having lost connection with its previous master; at this point it resets its clock, starts to increment it and sends out a 10 KHz synchronization clock of its own.

It's also possible to fake a master by generating a clean 10 KHz clock (50% duty cycle) and a 200 µs reset pulse and feeding these into the first device in the chain, by using external hardware such as a programmable function generator.

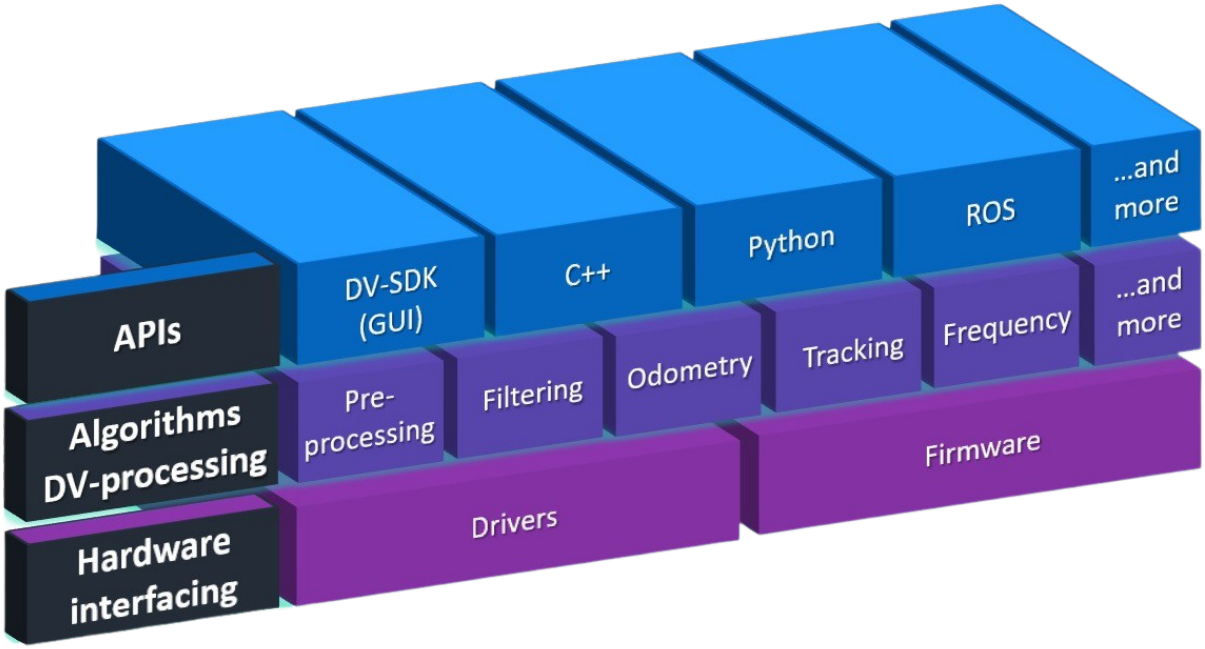
CHAPTER

FOUR

INTRODUCTION

We offer different software solutions to use our event cameras. The different software solutions are designed to be versatile and range from getting familiar with event cameras to *advanced usage*.

- *DV* is our main software solution and is designed to be user-friendly and easy to use.
- *DV-Processing* is a C++/Python library for generic processing algorithms.
- *Other software* solutions include a ROS integration and the low-level library Libcaer.



4.1 Common Definitions

Throughout our software stack, we keep common definitions.

4.1.1 Events

Events are the main data points generated by our cameras. In literature and, more particularly, in our software, they are defined as follows:
An `Event` is a single-pixel detection of brightness change. It is generally composed of the following fields:

- `timestamp`, represents the time when the change happened. It is represented as a Unix Timestamp[70] in microseconds. Type: `int64`
- `x`, represents the horizontal coordinate on the sensor (in pixels) where the change happened. Type: `int16`
- `y`, represents the vertical coordinate on the sensor (in pixels) where the change happened. Type: `int16`
- `polarity`, represents the sign of the change that happened. `1` represents an increase in intensity, `0` represents a decrease in intensity. Type: `bool`

Implementation in dv-processing[71]

4.1.2 Frames

Frames are the containers for image data.
A `Frame` is generally composed of the following fields:

- `timestamp` represents the time of the start of exposure of the frame. It is represented as a Unix Timestamp[72] in microseconds. Type: `int64`
  - There can be other timestamp fields representing *start of exposure*, *end of exposure*, *start of generation* or *end of generation*.
- `pixels` represents the image data stored as a pixel array. It is based on OpenCV Mat types[73]. Type: array of `uint8`
- `sizeX` represents the image's width in pixels. Type: `int16`
- `sizeY` represents the image's height in pixels. Type: `int16`
- `positionX` represents the horizontal image offset from the upper-left corner of a sensor in pixels. Type: `int16`
- `positionY` represents the vertical image offset from the upper-left corner of a sensor in pixels. Type: `int16`

Implementation in dv-processing[74]

4.1.3 Triggers

Triggers are data points forwarded by a camera informing about the waveform of an *external input signal*.
A `Trigger` is generally composed of the following fields:

- `timestamp`, represents the time when the signal component was detected. It is represented as a Unix Timestamp[75] in microseconds. Type: `int64`
- `TriggerType`, represents the signal component that was detected. For example, it can be one of `EXTERNAL_SIGNAL_RISING_EDGE`, `EXTERNAL_SIGNAL_FALLING_EDGE`. Type: `int8`

Implementation in dv-processing[76]

4.1.4 Accumulation

Accumulation is the process of reconstructing images from events by integrating the individual event values over time. This term is often used instead of *Image Reconstruction*.

Read more in the *section about the accumulator module*.

[70] https://www.unixtimestamp.com/
[71] https://gitlab.com/inivation/dv/dv-processing/-/blob/master/include/dv-processing/data/event.fbs
[72] https://www.unixtimestamp.com/
[73] https://docs.opencv.org/4.8.0/d3/d63/classcv_1_1Mat.html
[74] https://gitlab.com/inivation/dv/dv-processing/-/blob/master/include/dv-processing/data/frame.fbs
[75] https://www.unixtimestamp.com/
[76] https://gitlab.com/inivation/dv/dv-processing/-/blob/master/include/dv-processing/data/trigger.fbs

CHAPTER

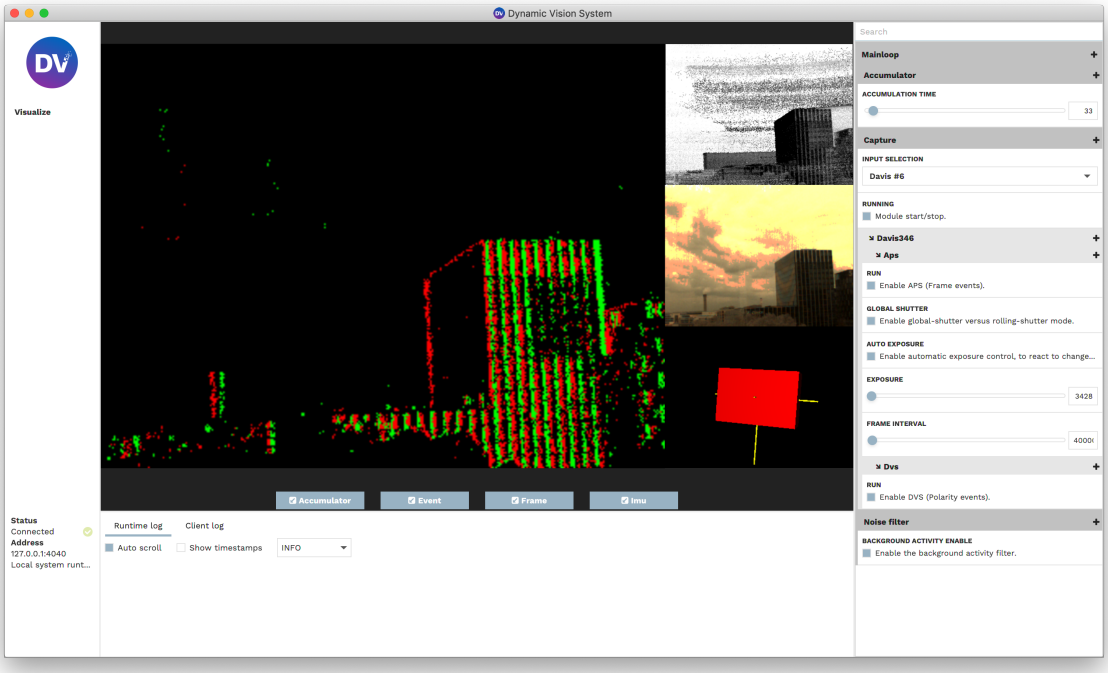
FIVE

DV

We provide an open-source software called 'DV' To visualize, record and process the output data from DVS cameras. It is designed to streamline the integration of DVS camera data into various computer vision applications, making it a valuable tool for researchers and developers in the field. Its user-friendly interface and compatibility with popular programming languages make it accessible to a wide range of users.

It is composed of two main components:

- The *GUI*, that allows users to easily interact with our cameras, process and visualize their data.



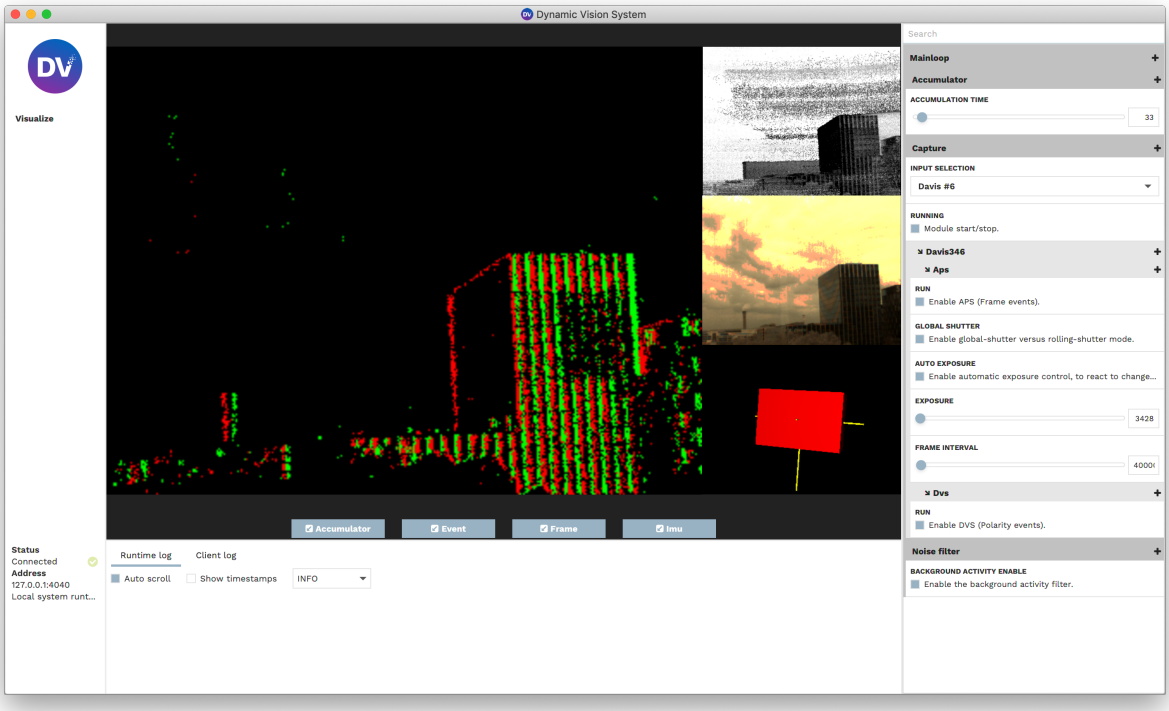
- The *runtime*, the backend of the GUI that takes care of all the processing logic. It is controlled via the GUI.

Both of these components are made to work with *DV Modules*.

Along with this software, we provide a library, *dv-sdk*, that allows users to extend the provided functionalities of the software or integrate their algorithms into it.

5.1 DV-GUI

The GUI of our software is a user interface that allows users to easily interact with our cameras, process and visualize their data.



A detailed introduction on how to use it is presented in this documentation.

5.1.1 Install

DV supports Windows, macOS and Linux. We provide pre-built packages for the most common operating systems. You can find instructions for installing DV on your operating system below.

Download and install the latest DV release for the respective operating system from the link below.

Windows

Download for Windows (Intel 64-bit)[77]

MacOS

MacOS (ARM 64-bit)

Download for macOS (ARM 64-bit)[78]

[77] https://release.inivation.com/gui/latest-win-stable
[78] https://release.inivation.com/gui/latest-mac-arm-stable

Linux

Please note that the GUI version of our DV software is only available on x86_64 architectures, for ARM-based devices please check the *Embedded Devices* section.

Ubuntu Linux

We provide a PPA repository[79] for Focal (20.04 LTS), Jammy (22.04 LTS), Noble (24.04 LTS) and Resolute (26.04 LTS) on the x86_64 and arm64 architectures.

```
sudo add-apt-repository ppa:ubuntu-toolchain-r/test
sudo add-apt-repository ppa:inivation-ppa/inivation
sudo apt-get update
sudo apt-get install dv-gui
```

Fedora Linux

We provide a COPR repository[80] for recent Fedora releases on the x86_64, arm64 and ppc64le architectures.

```
sudo dnf copr enable inivation/inivation
sudo dnf install dv-gui
```

Arch Linux

You can find DV in the AUR repository, install the package dv-gui[81].

Gentoo Linux

A valid Gentoo ebuild repository is available here[82] over Git. The package to install is `dev-util/dv-gui`.

Embedded Devices

Our GUI software is not supported on embedded devices. However, such devices can be used to run remote instances of `dv-runtime` to be accessed by a *remote GUI*. For that reason, it is still possible to install dv-runtime on its own[83].

Bleeding edge releases

We provide a bleeding edge release with the latest features directly from development. **Bleeding edge releases are untested and are potentially broken.** Use at your own risk or when directed to do so.

- Latest Git master Windows build (64-bit)[84]
- Latest Git master macOS build (ARM 64-bit)[85]

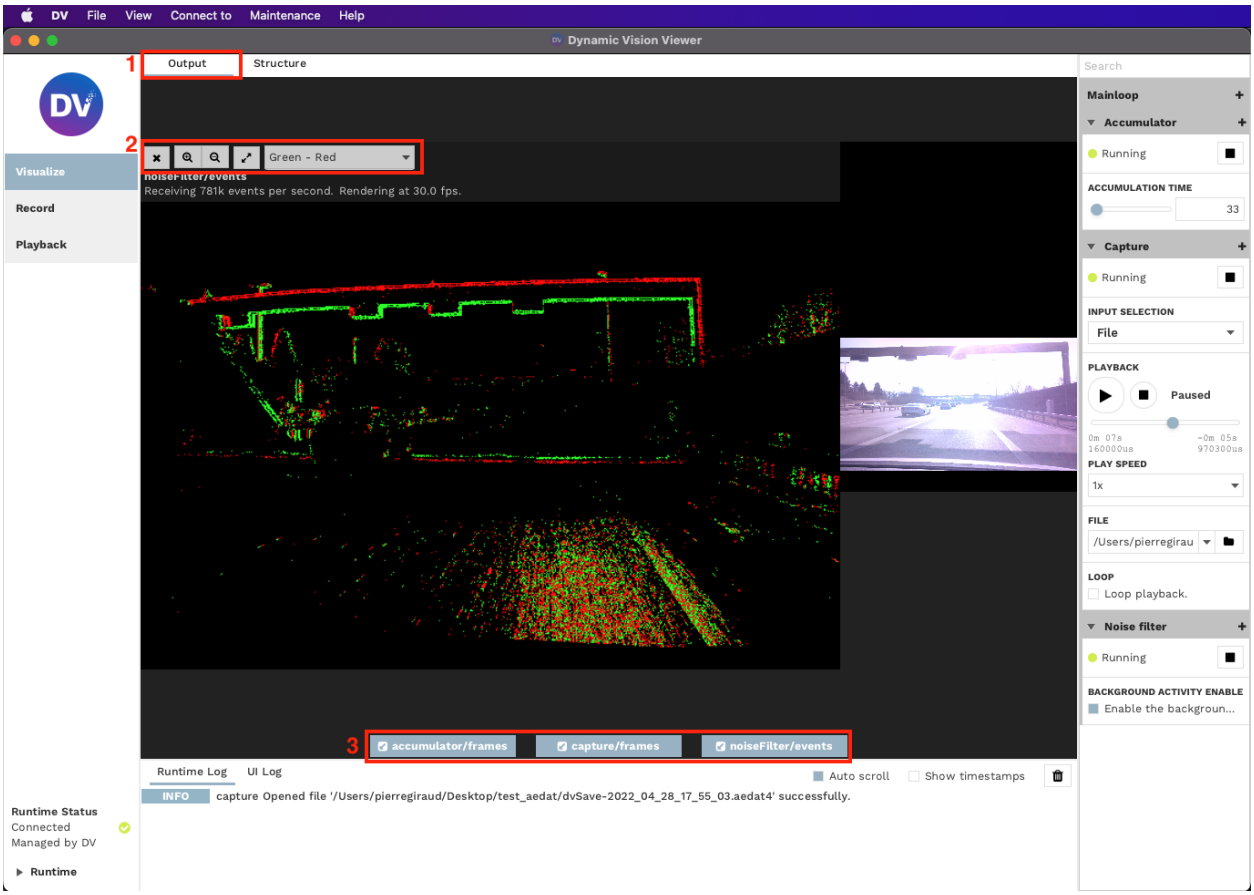
[79] https://launchpad.net/~inivation-ppa/+archive/ubuntu/inivation
[80] https://copr.fedorainfracloud.org/coprs/inivation/inivation/
[81] https://aur.archlinux.org/packages/dv-gui
[82] https://gitlab.com/inivation/gentoo-inivation/
[83] https://dv-runtime.inivation.com/master/runtime/installation.html
[84] https://release.inivation.com/gui/latest-win-master
[85] https://release.inivation.com/gui/latest-mac-arm-master

5.1.2 Getting Started

This page presents a beginner's introduction on how the GUI of our DV software is structured and how to use it.

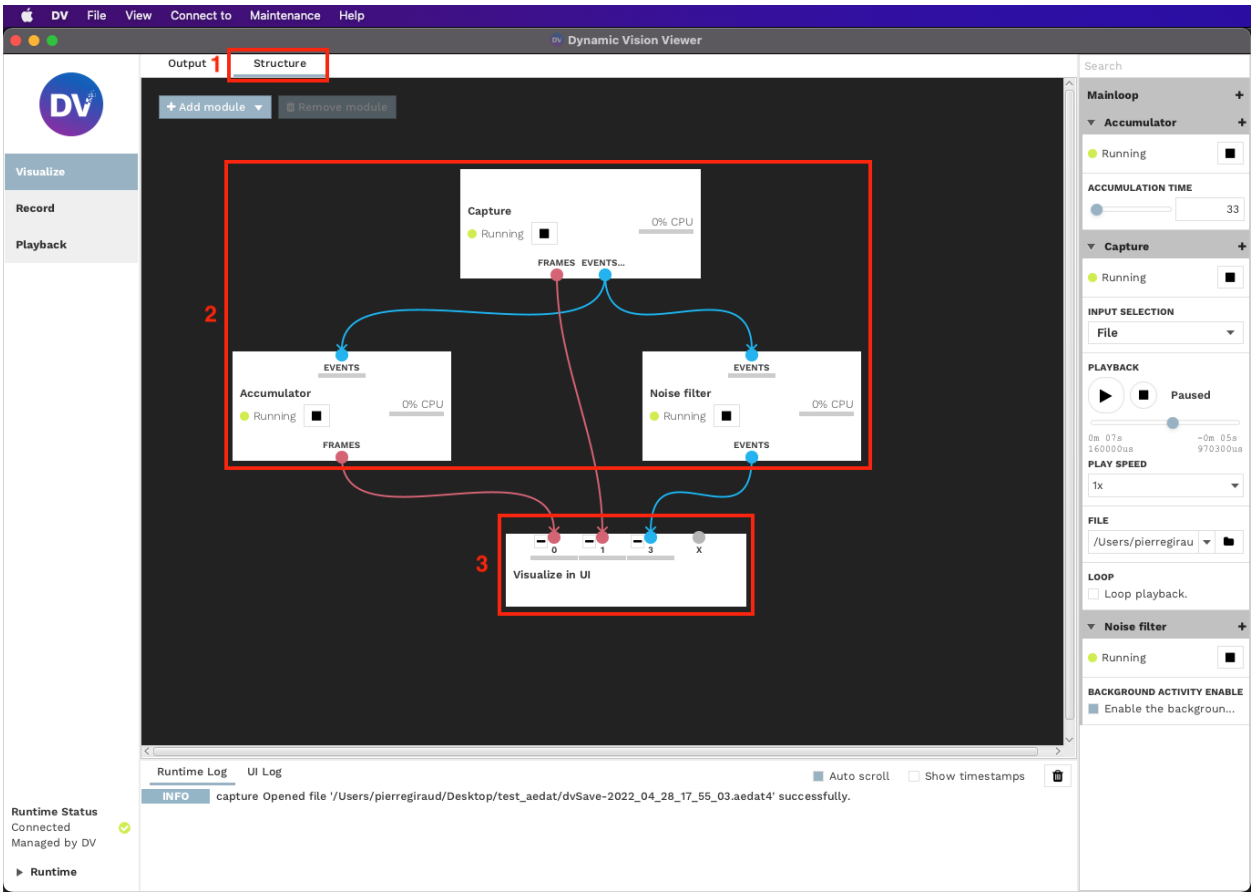
Output Tab

1. You can switch to the output tab of DV to visualize the output data.
2. You can control the windows for each data that is visualized (size, full screen, event color coding).
3. You can hide/show the different windows.



Structure Tab

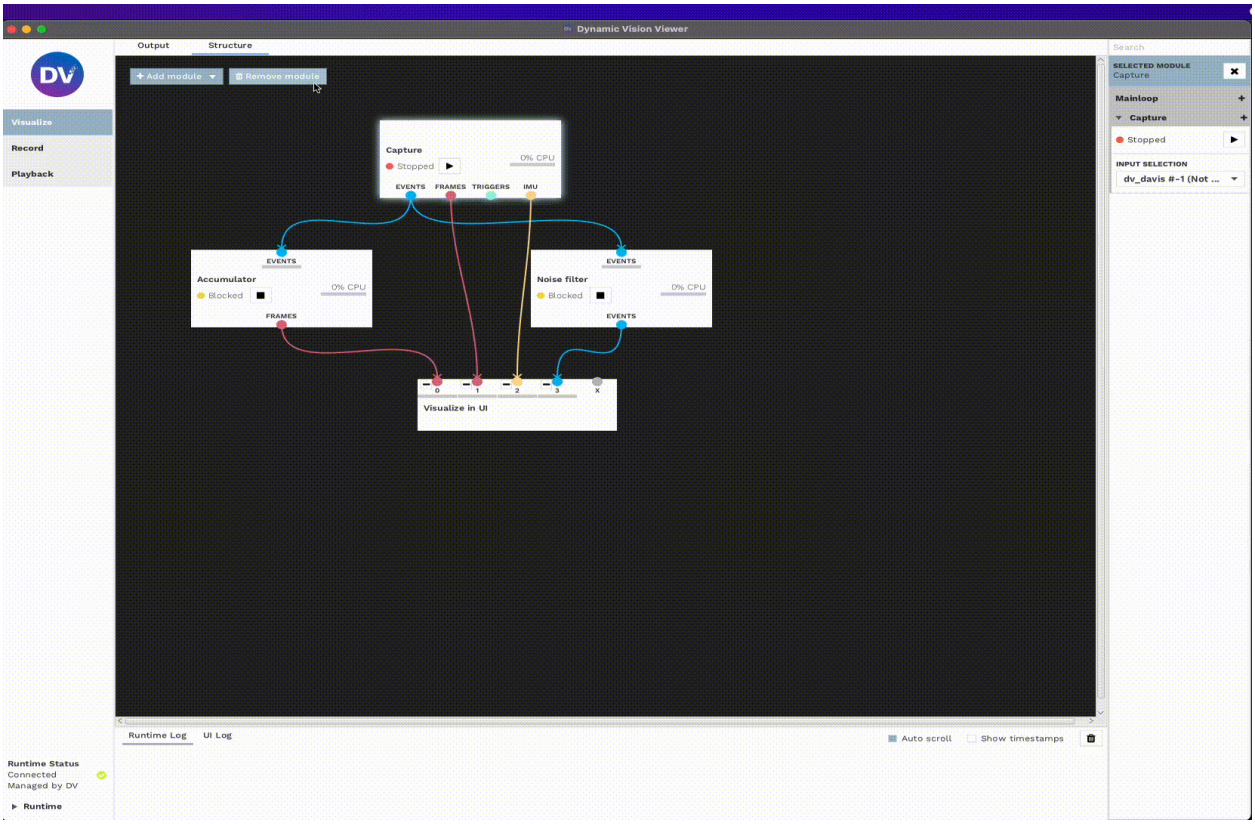
1. You can switch to the structure tab of DV to visualize the block logic (*DV Modules*) of the data processing.
2. You can view and control what processing is being performed.
3. You can see and select which data is being visualized (sent to the *Output Tab*)



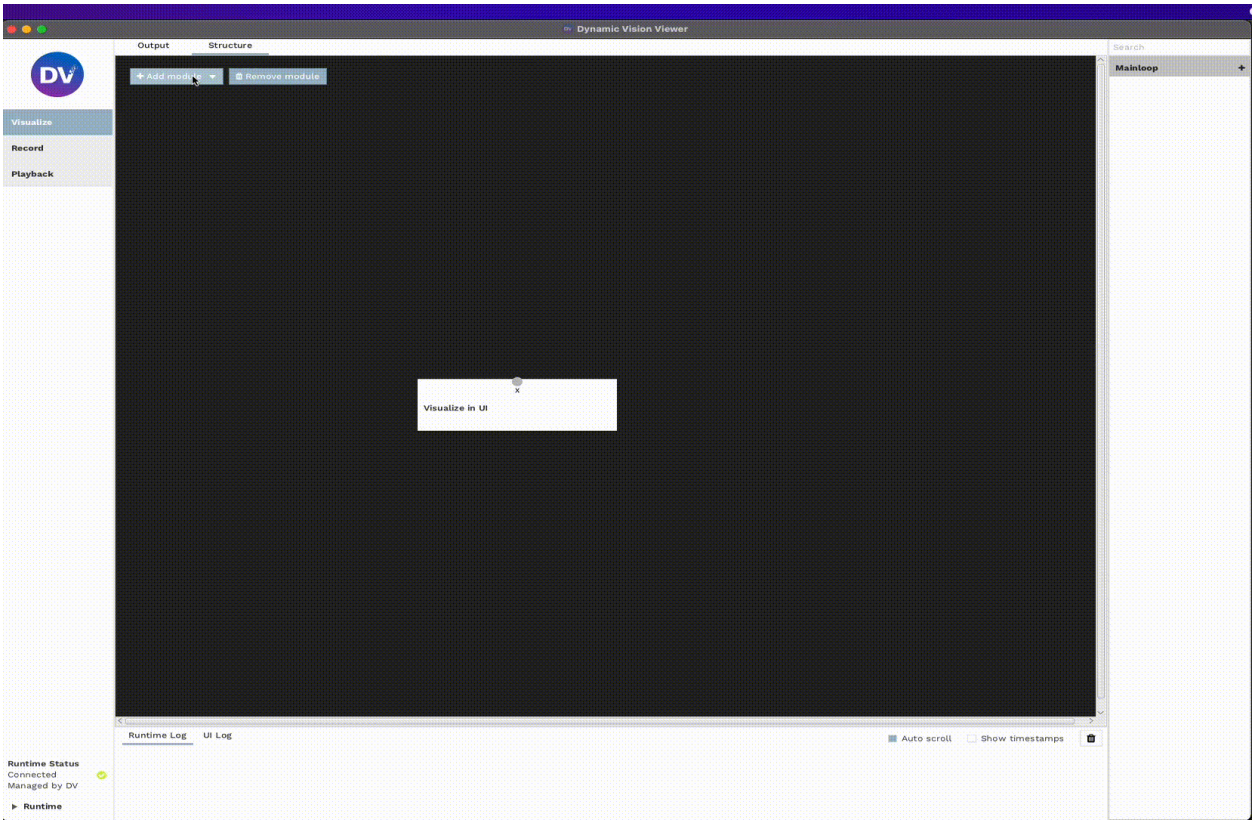
To control the processing structure, there are 3 main functionalities around the *Modules*:

- *Removing Modules* from the structure.
- *Adding Modules* to the structure.
- *Connecting Modules* together and controlling the data flow.

Removing Modules

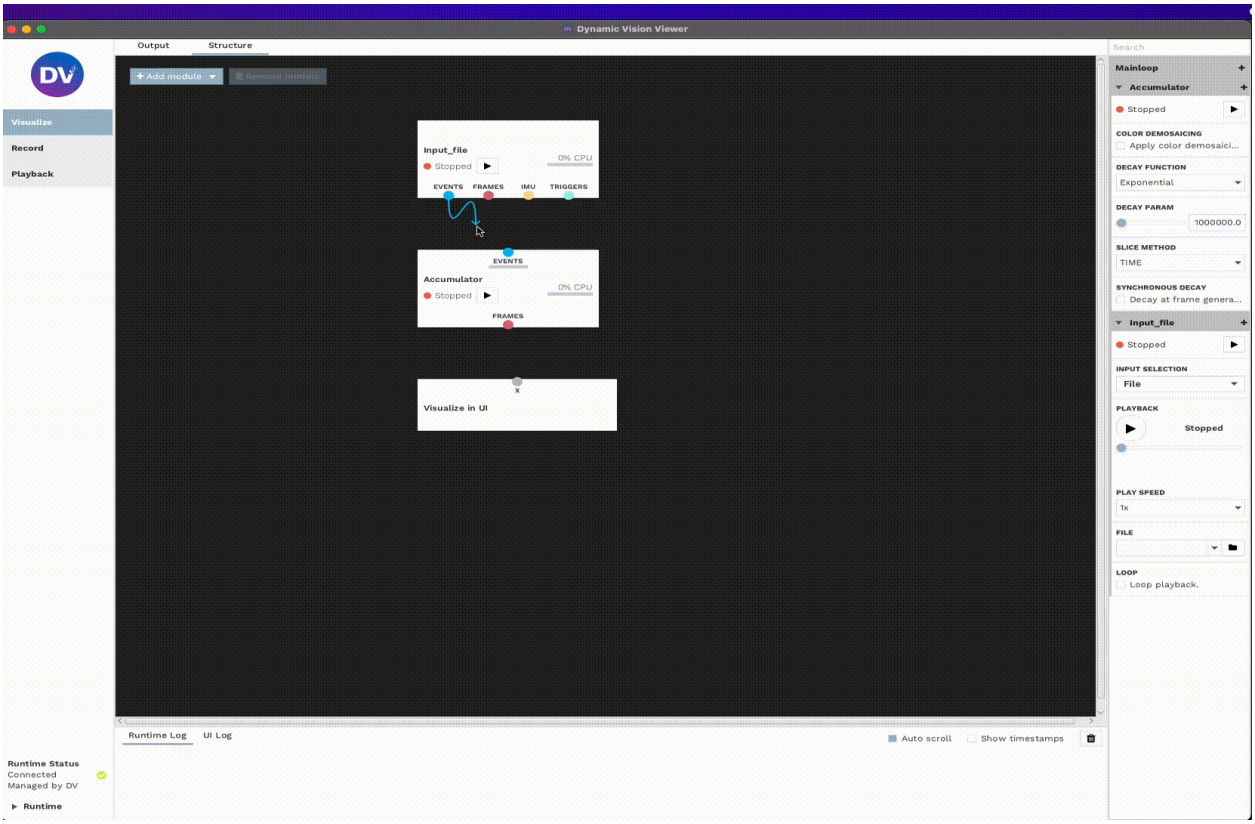


Adding Modules



Note: See the list of built-in modules in DV *here*

Connecting Modules



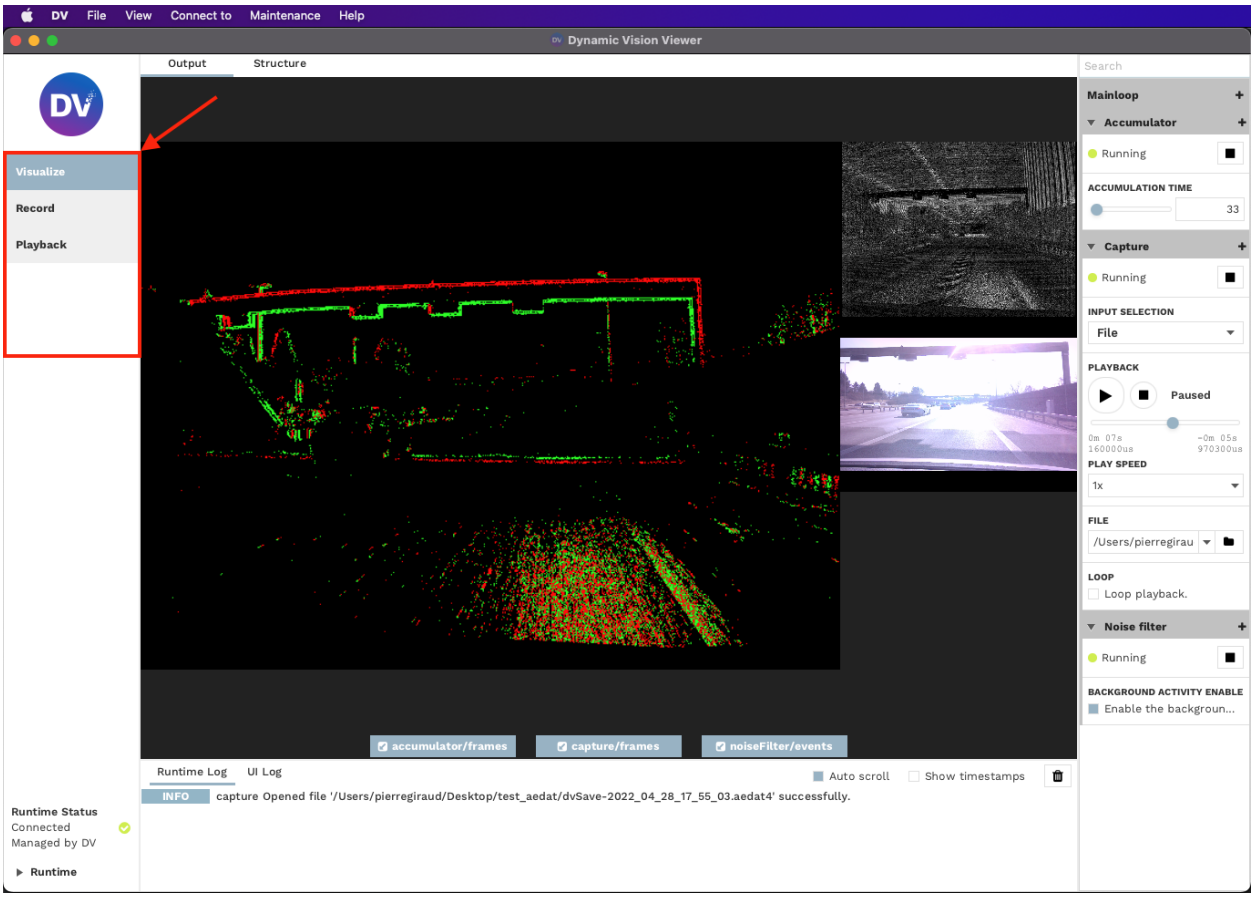
Project Bar

On the left side of `DV`, you have the `Project Bar`, where you can select existing or saved projects. These projects are `.xml` files that store the status of both the *Output Tab* and the *Structure Tab*.

Via the `File` tab at the very top of `DV`, you can:

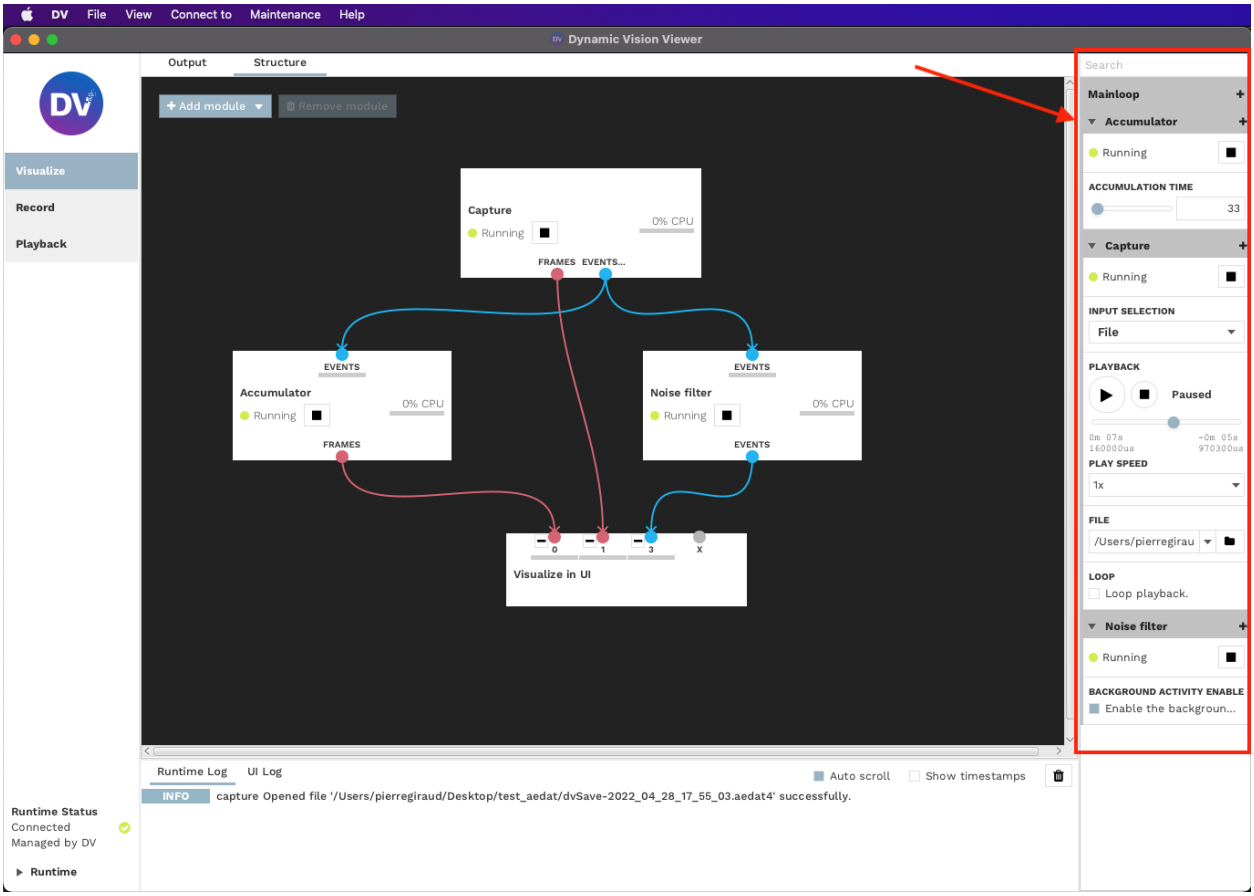
- Load existing projects with the `Open project` option.
- Create a new project from scratch with the `New project` option.
- Save the current status of your project via the `Save project` option.

The *Visualize*, *Record* and *Playback* are default projects that are always present in DV for convenience.

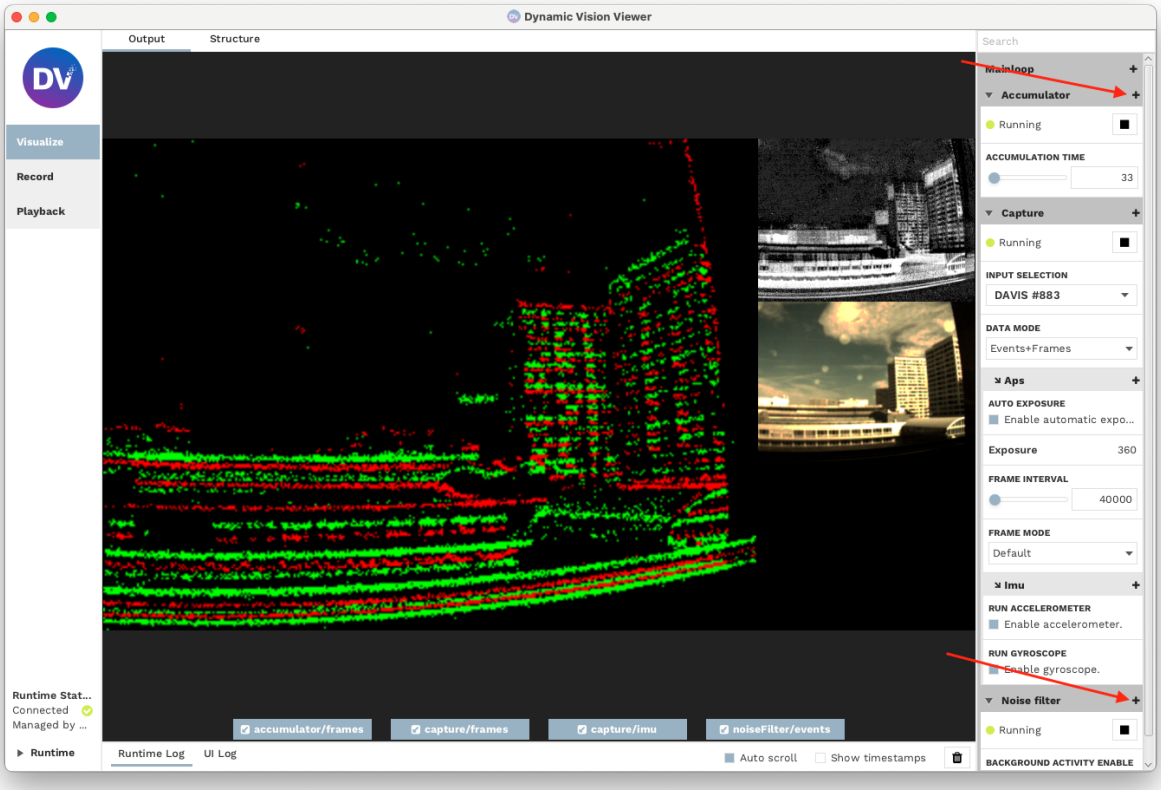


Configuration Bar

On the right side of `DV`, you have the `Configuration Bar`, this is where you can view and change the different parameters of the *modules* present in the *Structure Tab*.



Advanced settings can be shown by clicking the **+** button next to the module name.

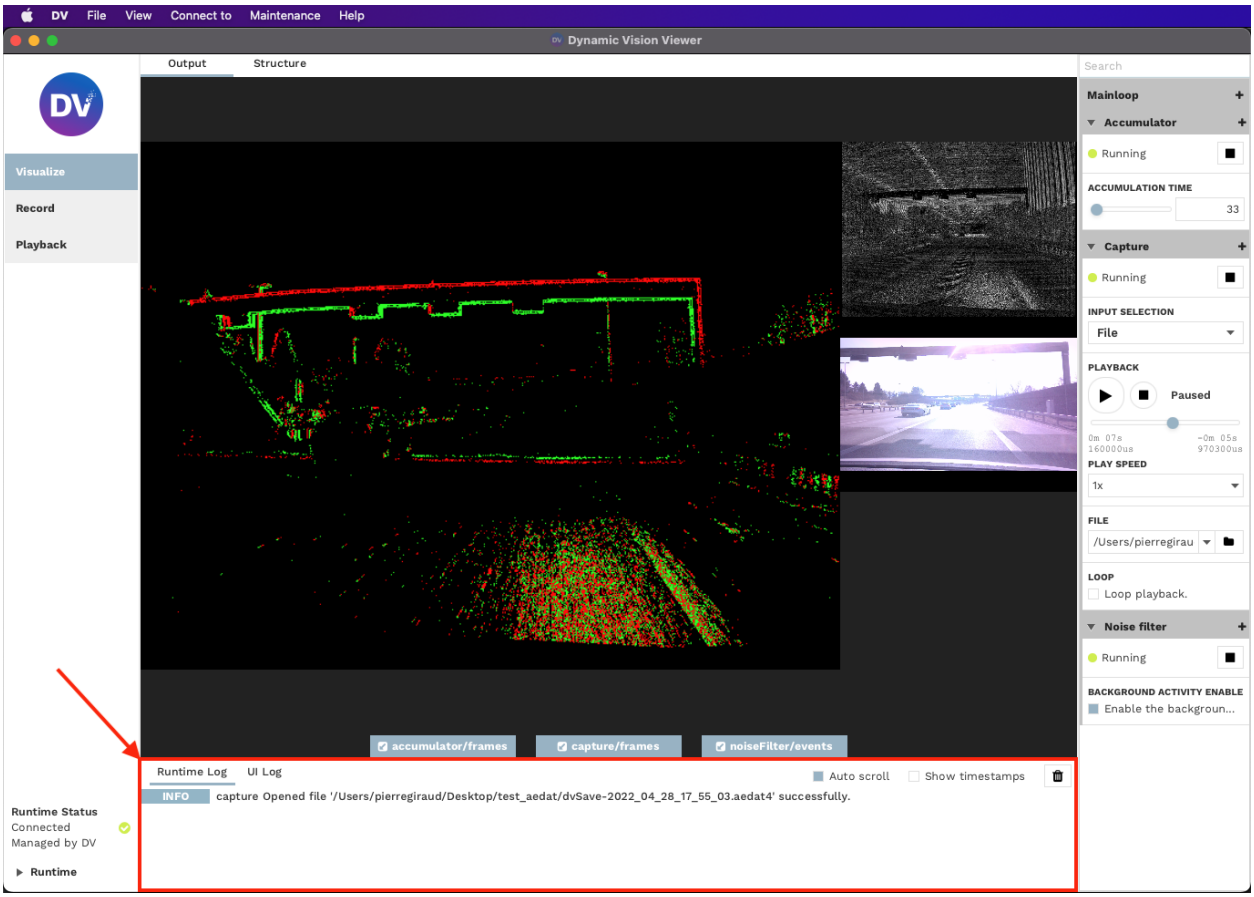


Logs/Console Bar

At the bottom of `DV`, you have the `Logs/Console Bar`, this is where any information or error message is displayed.

This section is split into two tabs:

- `Runtime Log` that will print messages related to the processing logic / modules.
- `UI Log` that will print messages related to the user interface, windowing, sizes, and buttons.



5.1.3 Visualize Data

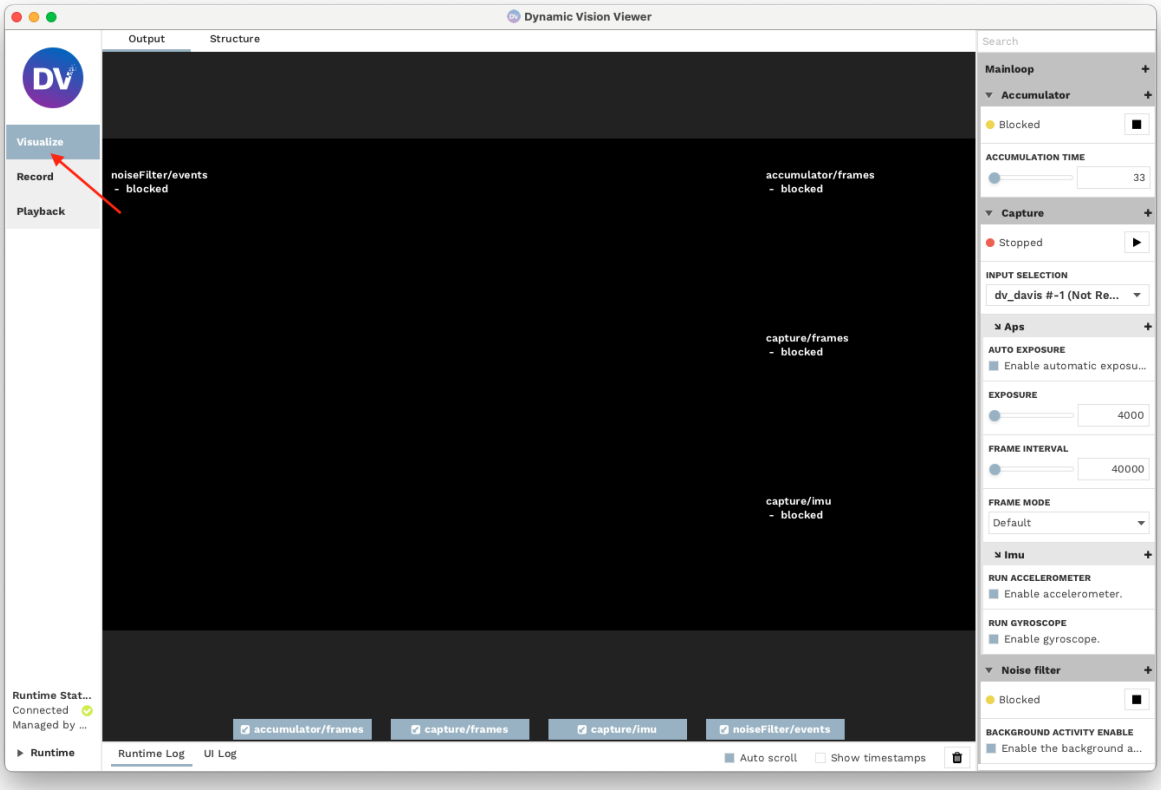
Starting DV

Start DV with

- **Windows** Double click the DV icon on the Desktop
- **macOS** Double click the DV icon in the Applications folder
- **Linux** Execute `dv-gui` in the command line or select `Development -> DV` in the system menu

Select Visualize Project

Select the *Visualize* project in the left sidebar of DV.



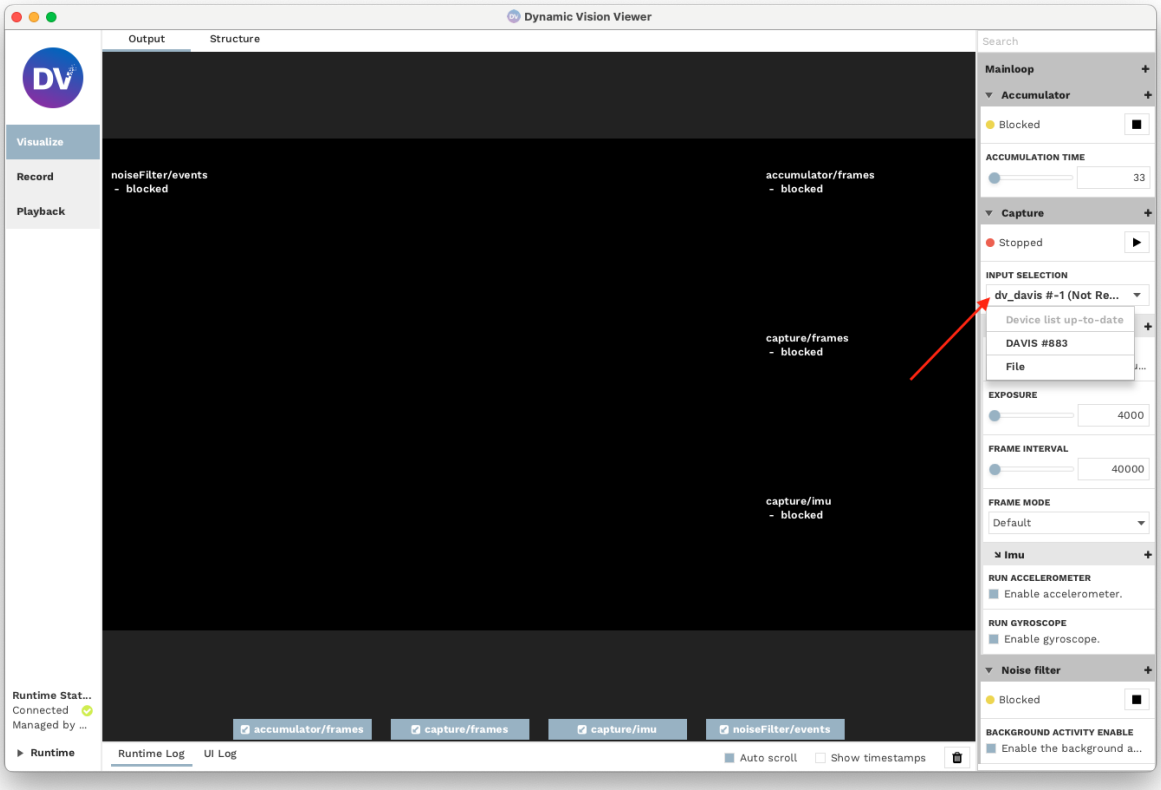
Select Input Device

Live Camera

Select the preferred camera to visualize in the input selection in the *Capture* settings. As highlighted in the Figure below, you can find this option on the right side of the DV window. The visualization should start as soon as you select the correct camera.

Playback

If you want to visualize playback data from AEDAT4 format, select the *File* option and select the file you want to open. The visualization should start as soon as you select the correct file.



Adjusting Visualization Options

The default *Visualize* configuration comes with two software modules enabled by default:

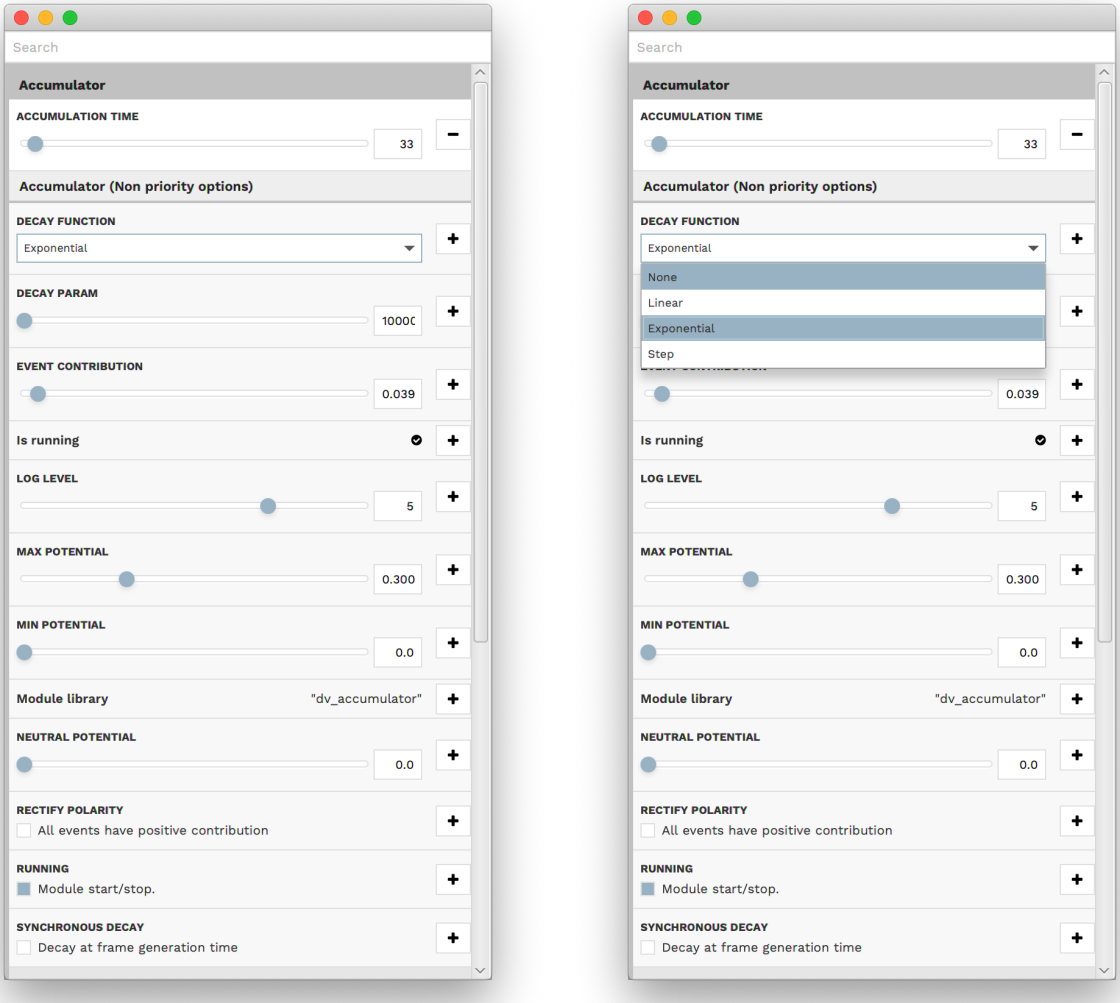
- A **Noise Filter** for the events output
- An **Accumulator** for basic frame reconstruction from events

The settings for both of these modules can be found in the *right-side configuration bar*.

Example: Disable Accumulation Decay

By default, the accumulator module decays the contribution of old events over time as soon as new events for a pixel arrive. This is, by default, done in an exponential manner.

To switch decay off, click the **+** button next to *Accumulator*.

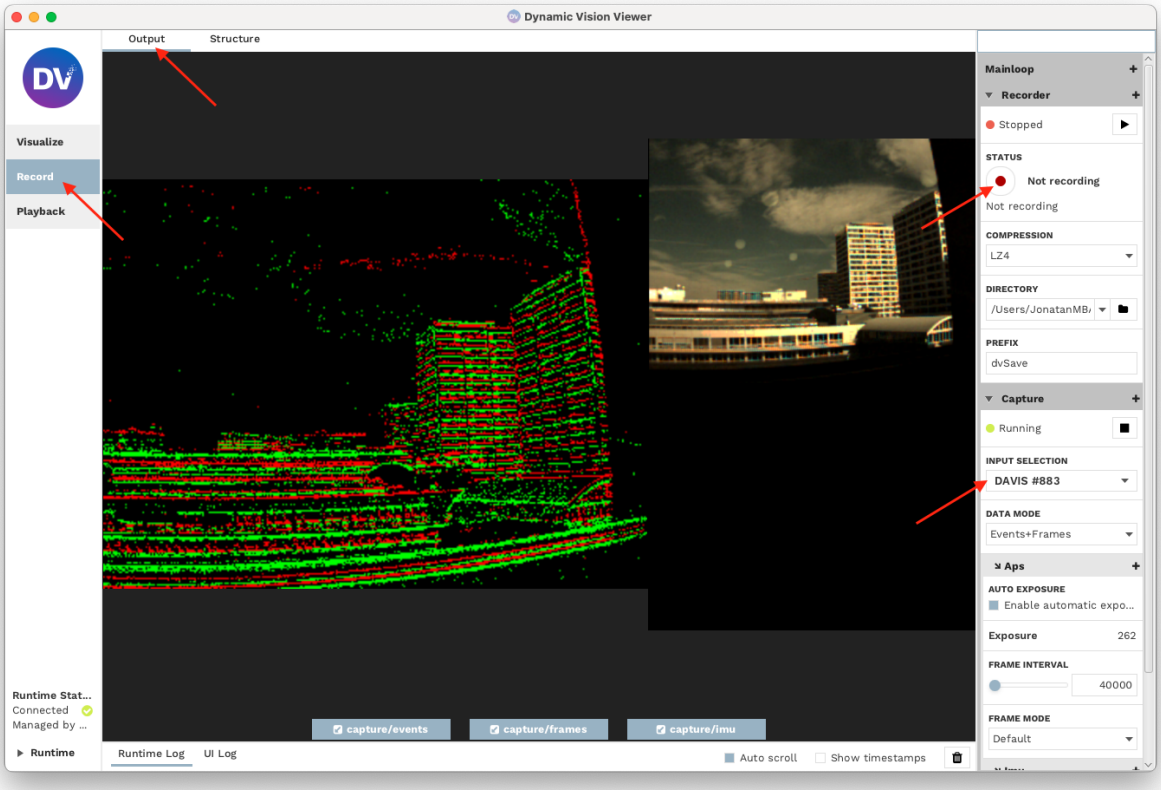


Switch the setting for the *Decay function* from *Exponential* to *None*. One can see that the accumulated image is a better representation of reality, but also accumulates more noise over time.

5.1.4 Record & Play Back Data

Record Data

1. Select the *Record* configuration in the left sidebar of DV. The current configuration gets replaced with a standard recording configuration. The default record configuration records all events, frames, imu, and trigger data from a connected DVS / Davis camera.
2. Select the *Output* tab to visualize the recording live.
3. Choose what camera input you want to record from under *Input Selection* in the *Capture* settings on the right side.
4. Recording can be started by clicking on the button with the red circle. By default, every recording creates a new file in your home directory. All data is recorded in the *aedat 4.0* format.



Play Back Data

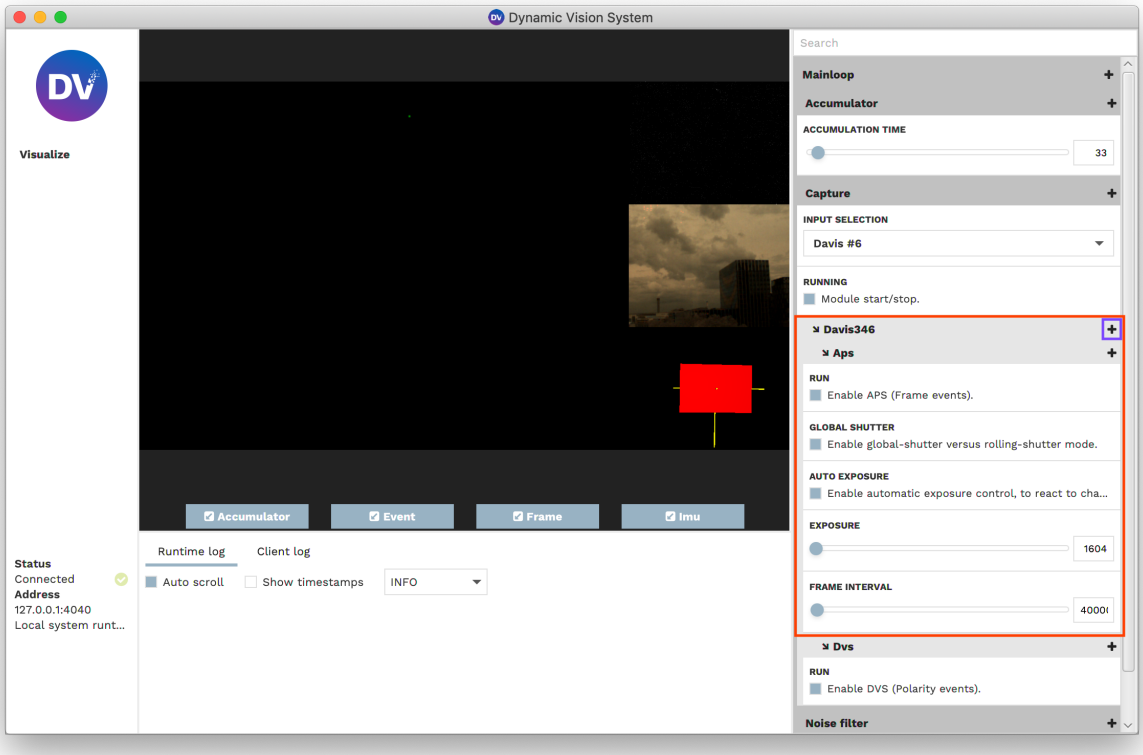
To play back and visualize the recorded data, see the tutorial on how to *visualize* the output.

5.1.5 Adjust Camera Settings

Priority Options

iniVation cameras have many settings that can be adjusted, like biases, exposure, global shutter, etc. The most important settings can be found in the configuration bar (on the right).

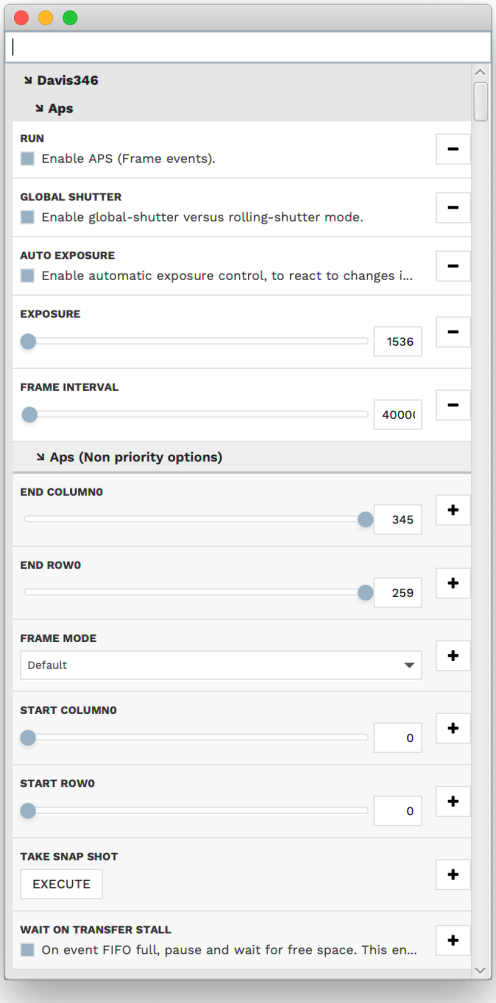
Note: different cameras have different sensor settings, see *hardware documentation* for more information.



Non-priority Options

All camera settings can be accessed by clicking on the **+** next to the camera name (shown with the blue annotation in the screenshot above).

A popup with all available settings appears. Any setting can be changed right from within this popup. Specific settings can be searched for by typing in the name of the setting in the search bar.





Add / Remove an Option to Priority Options

Any option that can be added to priority options has a **+** button to the right. Clicking this button adds the setting to the list of settings accessible from the right bar in the main window.

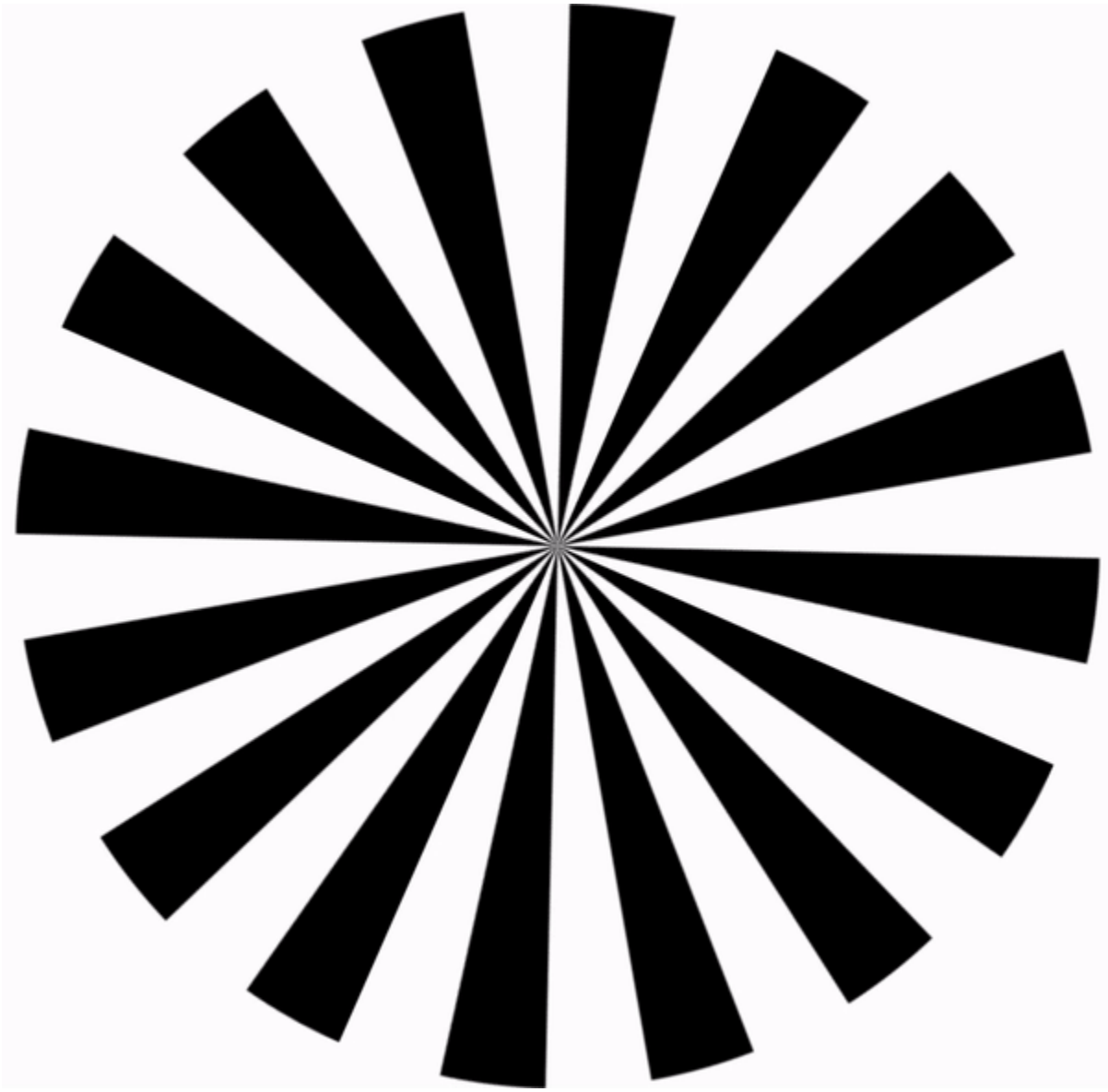
To remove an option from priority options, either click the **-** button to the right of the option or right-click on the option and select *Remove from priority option*.

The order of priority options can be changed by right-clicking on the option.

5.1.6 Focus an Event Camera

Focusing can be tricky with any camera, but if you don't have normal frames it will be even harder. In this small tutorial, you will learn how to focus an event-only camera properly.

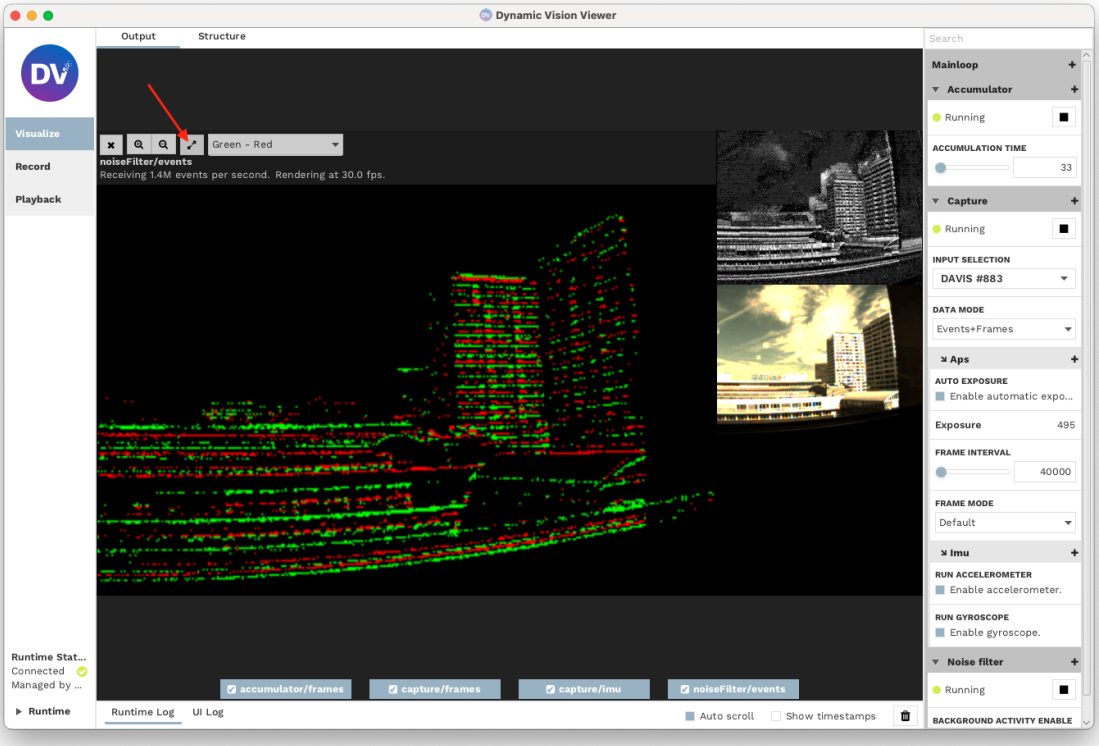
The easiest way to focus an event-only camera is using a Siemens Star.



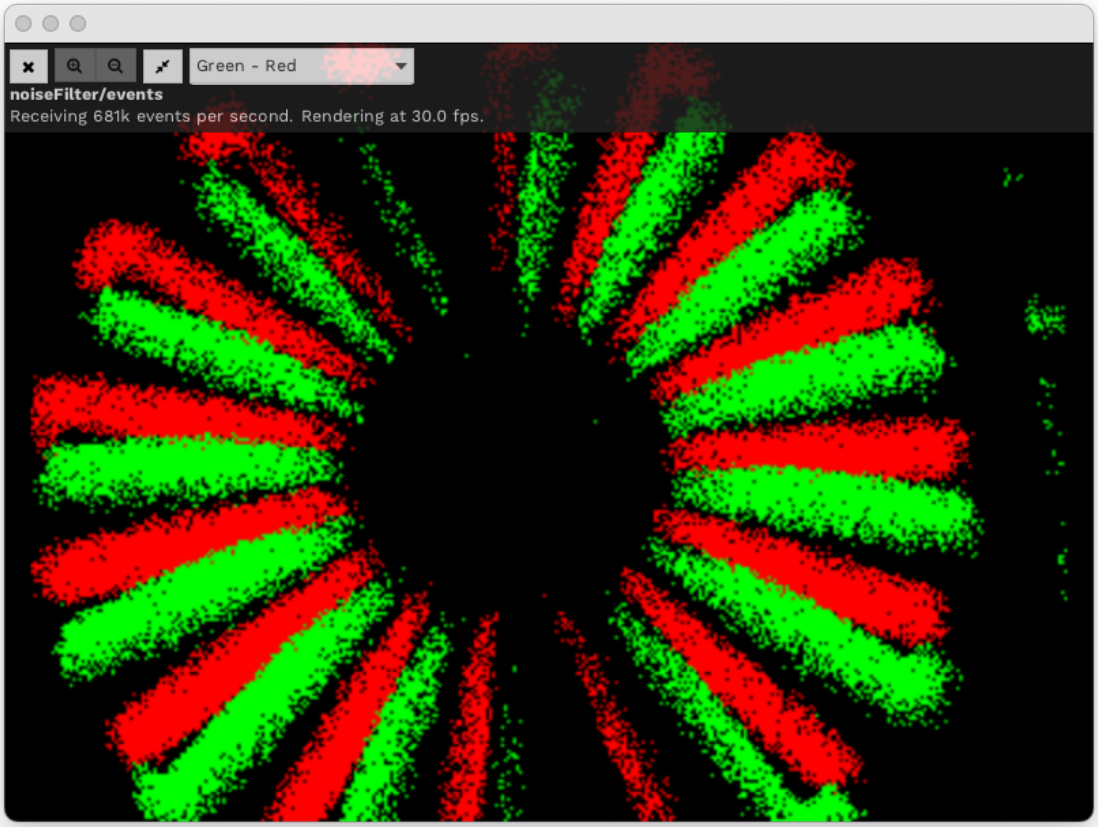
To focus an event camera, you can follow these steps:

1. Use a high resolution monitor to show a rotating Siemens Star as above. Alternatively, print the Siemens star (PDF[86]) on paper. Place the camera away from the Siemens Star, at the distance where you want to focus. The star should be clearly visible in the middle of the frame.
2. Start the DV software to visualize the camera, see how in the *visualize* tutorial.
3. Enlarge the event output visualizer or make it fullscreen by clicking the dedicated button.

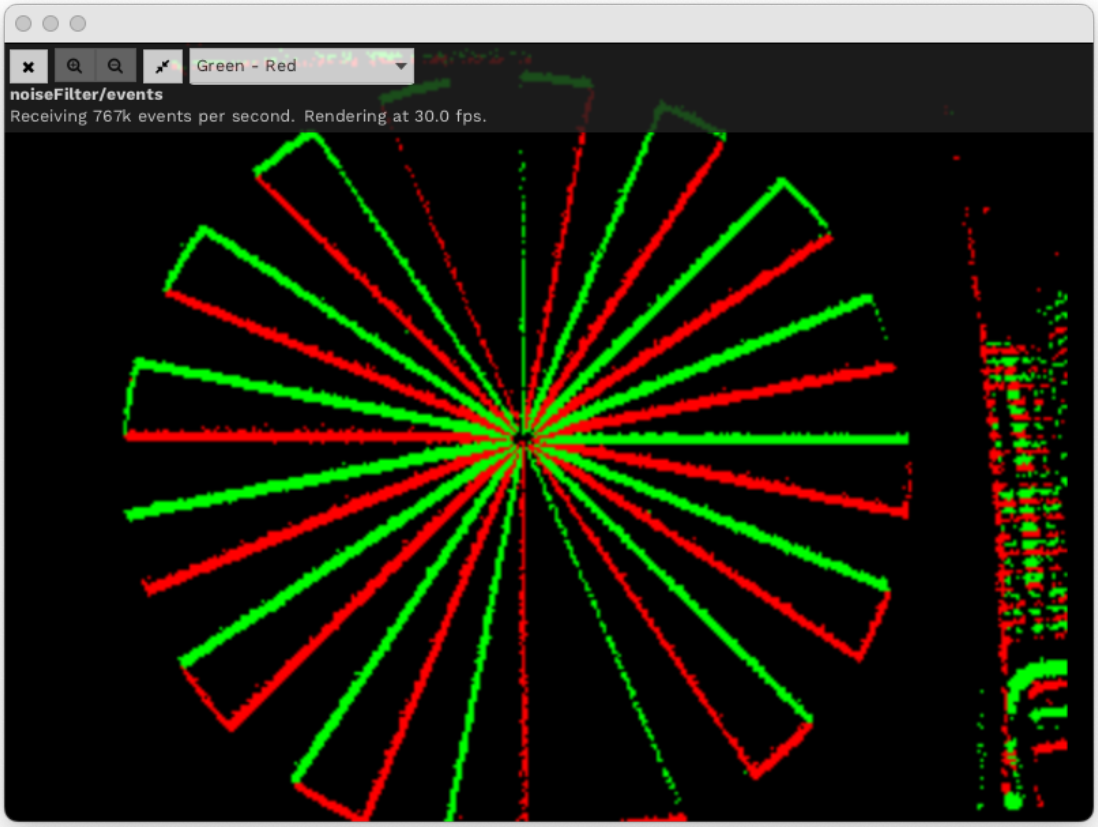
[86] https://docs.inivation.com/_static/calibration/siemens-star-24.pdf



4. Start to rotate the focus ring on the lens. If you have a paper print out of the Siemens star, keep generating events by slightly jiggling either the camera or the pattern. At first, if the camera is out of focus, you might see something like the image below.



5. Adjust the focus ring until the pattern's centre is visible and sharp. Below is an example of the output with a focused



camera.

Note: For a bigger depth of field (focus range), you need to have a higher f-number, which will also reduce the amount of light the sensor will receive.

5.1.7 Calibrate an Event Camera with DV

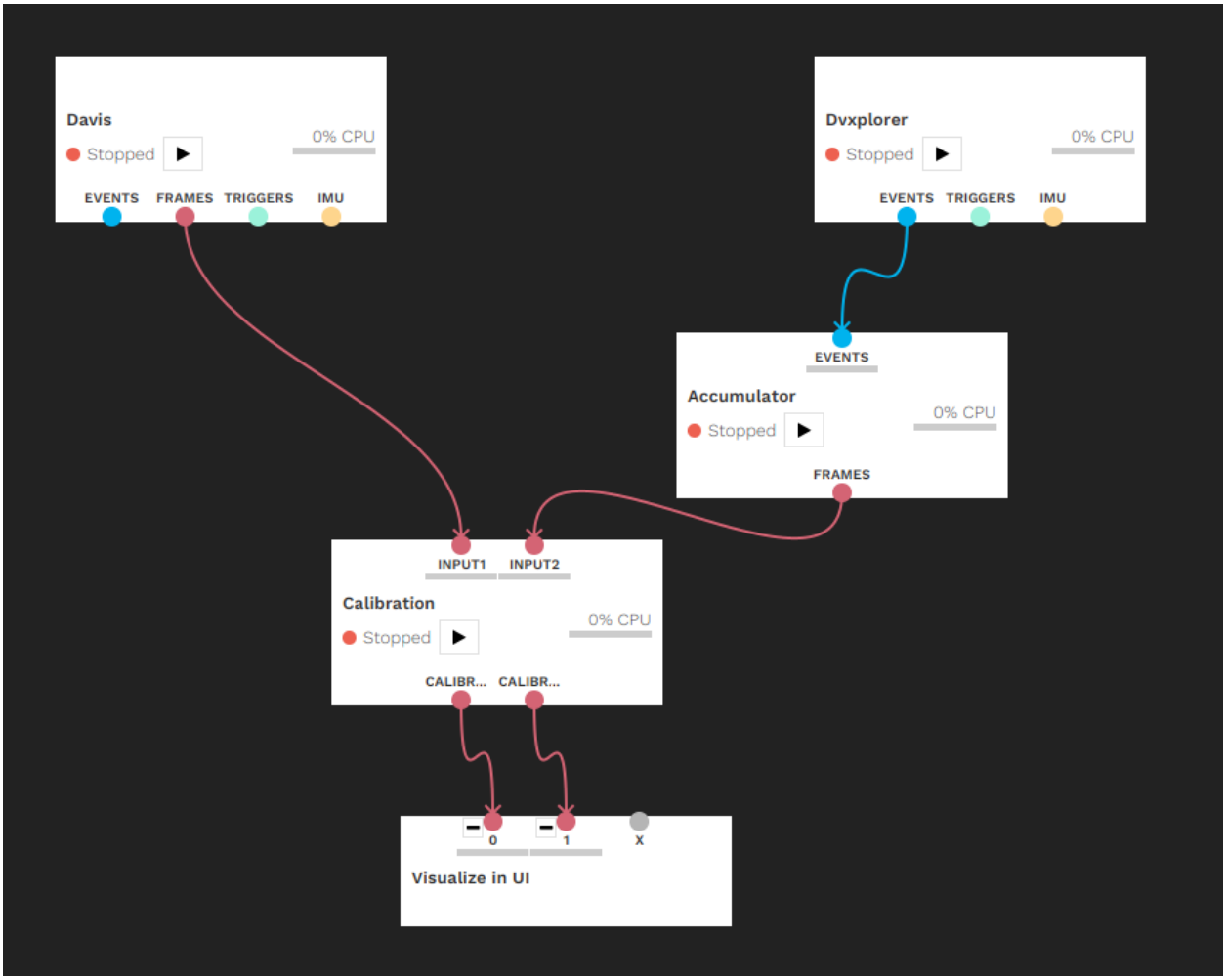
Prerequisites

- One or two cameras.
- Printed calibration pattern with known size.
- Synchronization cable in case of stereo calibration.

Setup

The mono or stereo calibration procedures are similar, and the following steps are valid for both scenarios.

The basic structure consists of the camera module(s) and the 'calibration' module. For each device without a frame output, such as the DVXplorer camera, an accumulator must also be added. A sample structure using a DAVIS 346 and a DVXplorer is shown below.



IMPORTANT: For stereo calibration, the two inputs need to be synchronized in time, for which a synchronization cable is required, and timestamps need to be reset from the master camera.

The following parameters must be set according to the calibration pattern: board height and width, square size and pattern type. The available patterns are the standard chessboard (Download PDF[87]), circles grid (Download PDF[88]) and asymmetrical circles grid (Download PDF[89]). In this tutorial, the 6 x 9 chessboard with a square size of 30 mm is used. So the width should be set to 9, the height to 6 and the square size to 30mm; these are already the default values. It should be noted that the square size can be expressed in any length scale.

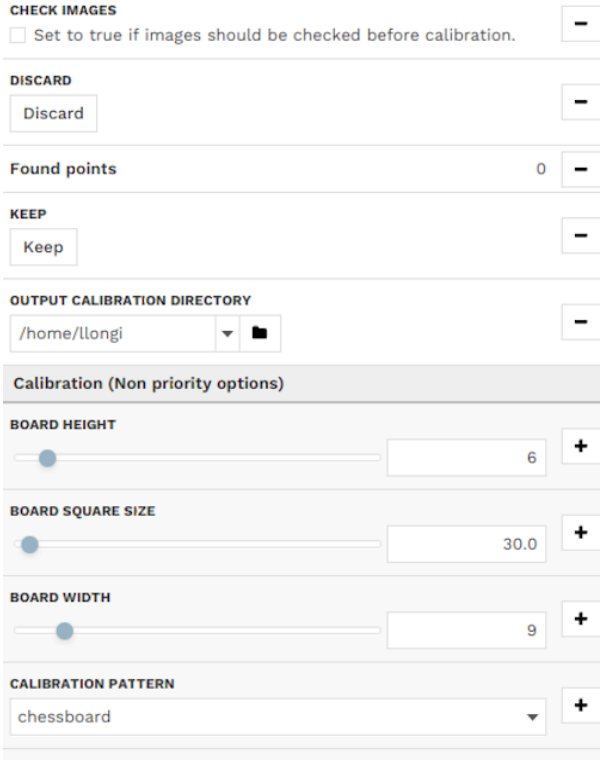
A few notes about calibration patterns:

- if you use our default patterns above, you can press the **Use Default Pattern** button in the configuration after selecting the appropriate pattern type, and it will automatically fill out all the sizes with the correct values.
- if you want to generate your own patterns, calib.io[90] has a very usable pattern generator, as well as printed patterns for sale.
- do not use square pattern sizes (i.e. 4x4, 8x8), nor symmetric ones (i.e. 6x8, 12x22), and use asymmetric patterns instead (one dimension even, one odd, i.e. 5x8, 10x19). This greatly helps calibration to figure out the proper orientation of the pattern and leads to better detections and results. A pattern size of at least 4x5 is recommended.

[87] https://docs.inivation.com/_static/calibration/calib.io-checker-279x210-6x9-30.pdf
[88] https://docs.inivation.com/_static/calibration/calib.io-circles-279x210-6x9-30-15.pdf
[89] https://docs.inivation.com/_static/calibration/calib.io-circlesA-279x210-9x12-28.2842-15.pdf
[90] https://calib.io/pages/camera-calibration-pattern-generator

- chessboard pattern: pattern width and height are the respective number of squares, and the square size is the size of one square's side.
- circles grid pattern: board width and height are the respective number of circles, and the square size is the distance between the centre of two consecutive circles.
- asymmetric circles grid pattern: width is the number of circles per row/column that stays constant, and height is the number of such groups. The square size is the distance between the centre of two consecutive circles on the same row/column.

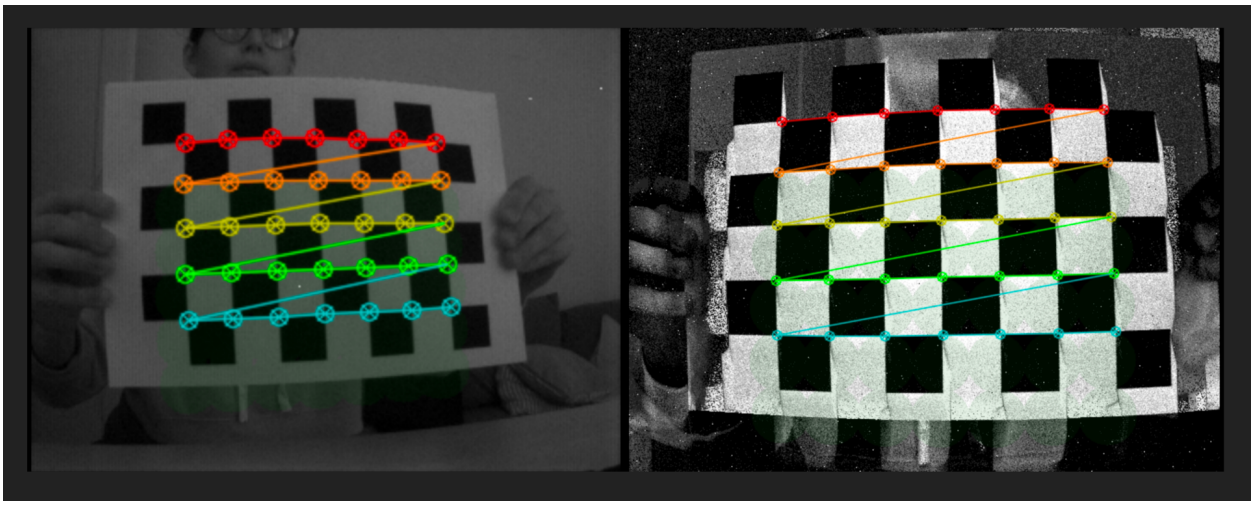
Additionally, an output directory can be defined under which the calibration results are saved. By default, or if the field is left empty, the home directory is used.



The calibration procedure is now ready to begin.

Calibrating

Click "run" on the calibration module. Start moving the calibration pattern in front of the camera(s). In case the pattern is detected, it will be drawn on the output image. The module collects a certain number of images for the actual calibration. This number can be set in the configuration panel under **Min Detections**. An image is added to the collection once the pattern has been consistently detected over a certain number of frames, configurable under **Consec Detects**. A log message will appear once an image has been added.



The area where a pattern has been found and added will turn slightly green. To get a good calibration result, one should cover the whole image in an even green shade. This can be turned off by unchecking the **Highlight Area** option.



Once enough images have been collected, they can be verified by the user if **Check Images** is ticked. One after another, the saved images and the detected patterns are shown. The user can choose to **Keep** or **Discard** them by pressing the corresponding buttons in the configuration panel. To get good calibration results, it is important to discard images where, although the pattern has been detected, some points are inaccurately placed or misaligned with the others or if too many images have been collected in a sub-region of the image.

New images are collected to replace the discarded ones, and once all pass the check, the calibration calculation begins. There may be some lag during the calculations, depending on the hardware and number of images used.

There are two error metrics. The maximum allowable error can be set under **Max Reprojection Error**. For stereo calibration, additionally, the error resulting from the epipolar line constraint is calculated and can be set under **Max Epipolar Line Error**. For the stereo calibration, both tests need to pass.

The epipolar line error is calculated as the mean epipolar error for every point in all the collected image pairs. For each pair of images, the error is computed as the sum of the distances *d()* between the points in one camera and the epipolar lines calculated from the other camera (*m* is the number of acquired images and *n* the number of points).

$$Error_{epipolar} = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} d(P1_{i,j}, epipolarLine2_{i,j}) + d(P2_{i,j}, epipolarLine1_{i,j})}{m * n}$$

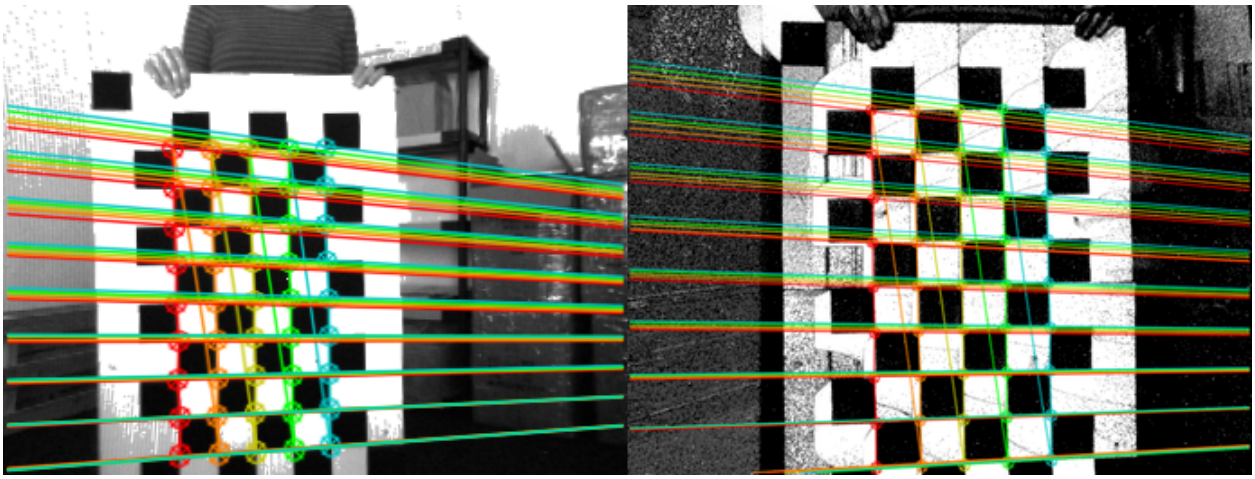
If the calibration is accurate enough, it is saved in the output directory, and the undistorted output from the camera(s) is shown. It is also possible to select the **Save Images** option to save the images used during the calibration in .png format.

If the calibration has been unsuccessful, it should be restarted with an increased number of images. To restart from the beginning with no stored images, simply stop and start the module again.

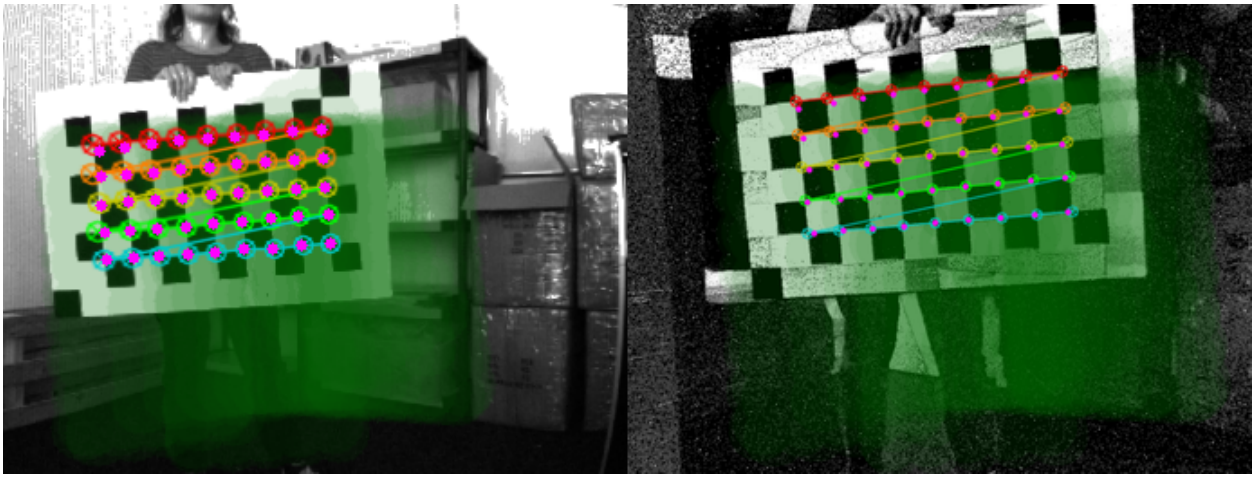
Additional options for stereo calibration

Calibrating two cameras with different resolutions requires both cameras to be calibrated separately first. The module tries to do this with the images provided to the stereo setup, however, this might not lead to good results for both cameras. In this case, both cameras should be calibrated independently using the calibration module with just one input. The saved results can be loaded for both inputs under **Input1 Calibration File** and **Input2 Calibration File** or alternatively from **Input Stereo Calibration File**.

In case the cameras have been successfully calibrated, epipolar lines can be drawn for the detected pattern if **Draw Epipolar Lines** is checked. This serves as a way to verify visually if the stereo calibration has been successful.



Another option for the stereo calibration is the possibility to draw the reprojected points from one camera to the other. You can select the **Draw reprojected points** option. This setting uses the Homography matrix (H) calculated during the calibration to map points from one camera to the other.



The option **Save Stereo Statistics** generates .csv files with the mean reprojected error of each acquired image from the two cameras.

Undistortion

Once calibration files have been generated for a camera, its event and frame outputs can be undistorted using the 'undistort' module. The module will try to verify that the data in the file applies to the connected camera inputs. Use one 'undistort' module instance per camera.

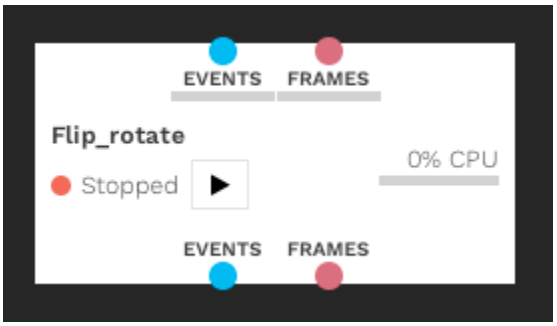
Improve calibration

- **ACCUMULATOR:** For calibrating cameras that have only events available it is important to tune the accumulation parameters. Rising **Event contribution** increases the image contrast and facilitates pattern detection (e.g. around 0.05).
- **ACCURACY:** For mono calibration, if the reprojection error is too high, it is possible to improve it by increasing the number of acquired images and ensuring that they cover the whole Field-Of-View of the camera. This is true also for stereo calibration and the epipolar error. In this latter case, calibration can also be improved by loading the single camera calibration matrices in **Input1 Calibration File** and **Input2 Calibration File** from previous mono calibrations.
- **SAVING:** The current implementation saves the calibration files only if the calibration errors satisfy the requirements. Increase the **Max Reprojection Error** or **Max Epipolar Error** if you want to always save the calibration results.

5.2 DV Modules

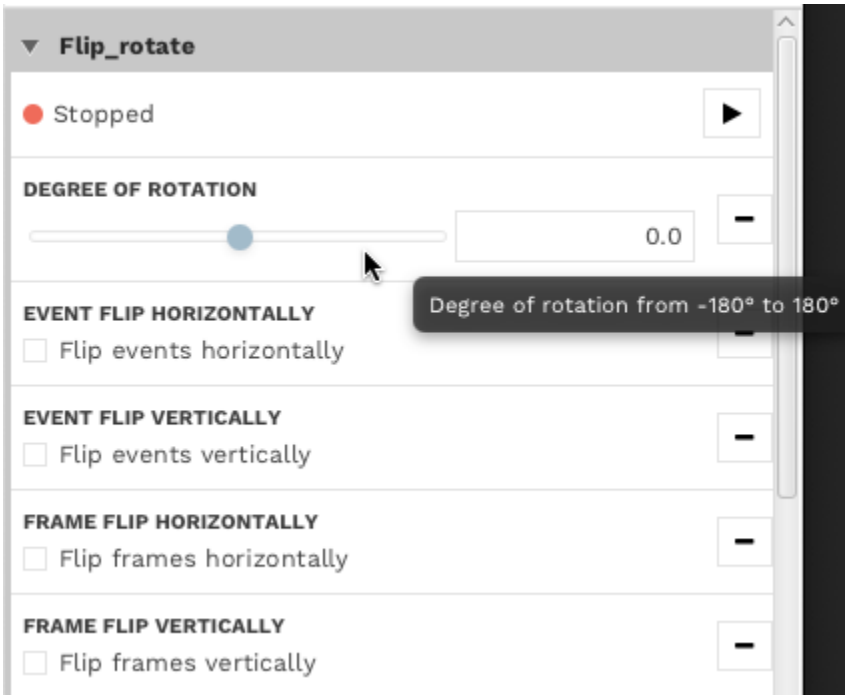
The `modules` in `DV` are logic blocks that are designed to perform a certain action with some data.

In DV, they are represented as follows.



- At the top: the colored dots, if any, represent the data input, the `Inputs`.
- At the bottom: the colored dots, if any, represent the data output, the `Outputs`.

They also have options that allow changing their logic (some options can even change the logic while running). These options can be viewed in the *Configuration Bar* of the GUI. Note that each option has a description that can be viewed by simply hovering the cursor over it.



In both locations, modules can be started with the button or stopped with the button.

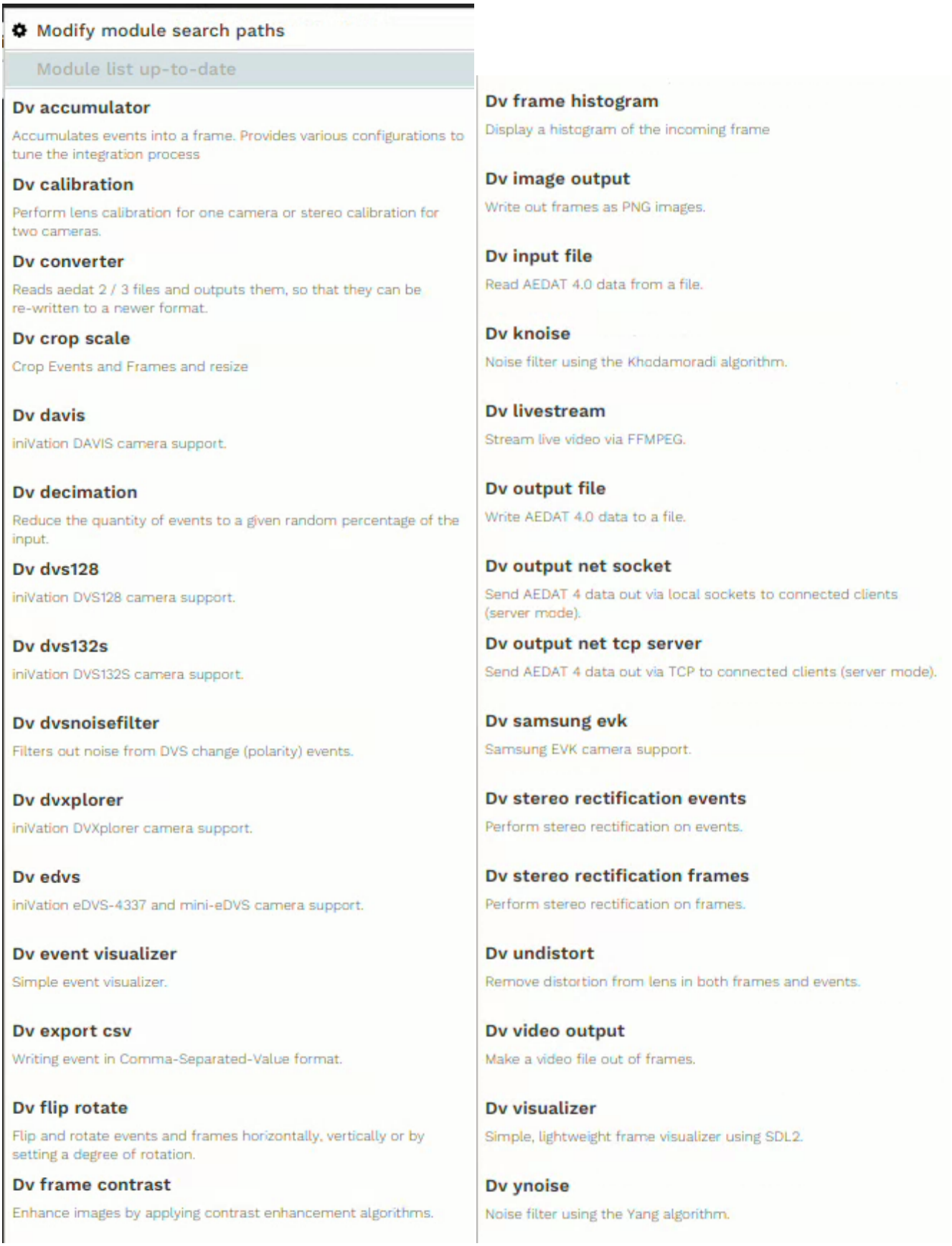
For more details about modules, see *the list of built-in modules in DV*.

It is also possible to create your own DV Module by using *dv-sdk*.

5.2.1 Built-In Modules

As part of the DV application, some *modules* are already provided to the users. Most of them provide basic features like camera or file input, but some of them can be more complex.

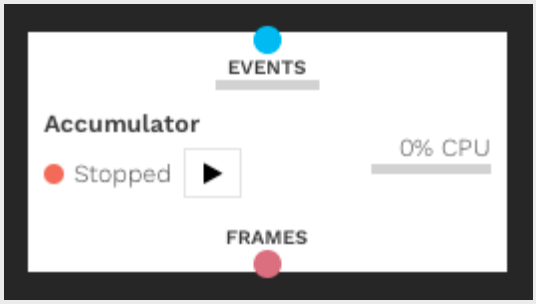
Here is a view of the list of modules as exposed in the *GUI*:


Modify module search paths
Module list up-to-date
Dv accumulator
Accumulates events into a frame. Provides various configurations to tune the integration process
Dv calibration
Perform lens calibration for one camera or stereo calibration for two cameras.
Dv converter
Reads aedat 2 / 3 files and outputs them, so that they can be re-written to a newer format.
Dv crop scale
Crop Events and Frames and resize
Dv davis
iniVation DAVIS camera support.
Dv decimation
Reduce the quantity of events to a given random percentage of the input.
Dv dvs128
iniVation DVS128 camera support.
Dv dvs132s
iniVation DVS132S camera support.
Dv dvsnoisefilter
Filters out noise from DVS change (polarity) events.
Dv dvxplorer
iniVation DVXplorer camera support.
Dv edvs
iniVation eDVS-4337 and mini-eDVS camera support.
Dv event visualizer
Simple event visualizer.
Dv export csv
Writing event in Comma-Separated-Value format.
Dv flip rotate
Flip and rotate events and frames horizontally, vertically or by setting a degree of rotation.
Dv frame contrast
Enhance images by applying contrast enhancement algorithms.
Dv frame histogram
Display a histogram of the incoming frame
Dv image output
Write out frames as PNG images.
Dv input file
Read AEDAT 4.0 data from a file.
Dv knoise
Noise filter using the Khodamoradi algorithm.
Dv livestream
Stream live video via FFMPEG.
Dv output file
Write AEDAT 4.0 data to a file.
Dv output net socket
Send AEDAT 4 data out via local sockets to connected clients (server mode).
Dv output net tcp server
Send AEDAT 4 data out via TCP to connected clients (server mode).
Dv samsung evk
Samsung EVK camera support.
Dv stereo rectification events
Perform stereo rectification on events.
Dv stereo rectification frames
Perform stereo rectification on frames.
Dv undistort
Remove distortion from lens in both frames and events.
Dv video output
Make a video file out of frames.
Dv visualizer
Simple, lightweight frame visualizer using SDL2.
Dv ynoise
Noise filter using the Yang algorithm.

Find more information about specific modules in their corresponding section.

Accumulator Module

Source code[91]

Entry in *Module Selection*	Module Appearance
Dv accumulator Accumulates events into a frame. Provides various configurations to tune the integration process	

What Does The Accumulator Do?

The accumulator module reconstructs frames from events by accumulating the changes to obtain a corresponding intensity information. The computation of the intensity is done with a configurable decay function.

Use cases:

- Create high-speed *event frames* (more details *here*)
- Frame reconstruction (more details *here* or *here*)
- Long term noise accumulation
- Creating representations to feed into a convolutional neural network

Accumulation can be used whenever there is a need to convert events into frames

Note: More info about accumulator implementation can be found in the dv-processing documentation[92].

Accumulator Settings Overview

Accumulation Time

Accumulation time sets the time window to accumulate events before sending out a new frame in *milliseconds*. This effectively sets the frame rate of the accumulator. The accumulator does not reset the frame at generation time, the contents of the old frame are carried over to the next frame. To achieve an effect where the frame gets cleared out after each frame, set the decay function to *Step* and make sure the *Decay param* is set to the same value as the accumulation time. Note that this value is in milliseconds, while the other is in microseconds.

Decay Function / Decay Parameter

One of *None*, *Linear*, *Exponential*, *Step*. Defines the data degradation function that should be applied to the image. For each function, the *Decay param* setting assumes a different function:

[91] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/accumulator
[92] https://dv-processing.inivation.com/master/accumulators.html

Function	*Decay param* function	Explanation
None	No function	Does not apply any decay
Linear	The slope `a` of the linear function, in *intensity per microsecond*	Assume the intensity of a pixel is `I0` at time `0`, this function applies a linear decay of the form of `I0 - (t * decayparam)` or `I0 + (t * decayparam)` until the value hits the value specified in *neutral potential*
Exponentia	The time constant `tau` of the exponential function in *microseconds*	Assume the intensity of a pixel is `I0` at time `0`, this function applies an exponential decay of the form `I0 * exp(-(t/decayparam))`. The decay approaches a value of `neutralPotential` over time.
Step	No function	Set all pixel values to *neutral potential* after a frame is extracted.

Event Contribution

The contribution an event has to the image. If an event arrives at a position `x`, `y`, the pixel value in the frame at `x`, `y` gets increased / decreased by the value of *Event contribution*, based on the events polarity.

Except:

- The resulting pixel value would be higher than *Max potential*, the value gets set to *Max potential* instead
- The resulting pixel value would be lower than *Min potential*, the value gets set to *Min potential* instead
- The event polarity is negative, and *Rectify polarity* is enabled, then the event is counted positively

Min Potential / Max Potential

Sets the minimum and maximum values a pixel can achieve. If the value of the pixel would reach higher or lower, it is capped at these values. These values are also used for normalization at the output. The frame the module generates is an `unsigned 8-bit` grayscale image, normalized between *Min potential* and *Max potential*. A pixel with the value *Min potential* corresponds to a pixel with the value `0` in the output frame. A pixel with the value *Max potential* corresponds to a pixel with the value `255` in the output frame.

Neutral Potential

Sets the neutral potential. This has different effects, depending on the decay function:

Function	*Neutral potential* function
None	No function
Linear	The value the linear function tends to over time
Exponential	No function
Step	The value the frame gets reset to after getting a frame

Rectify Polarity

If this value is set, all events act as if they had positive polarity. In this case, *Event contribution* is always taken positively.

Synchronous Decay

If this value is set, decay happens continuously for all pixels. In every frame, each pixel will be eagerly decayed to the time the image gets generated. If this value is not set, decay at the individual pixel only happens when the pixel receives an event. Decay is lazily evaluated at the pixel.

Note that both decay regimes yield the same overall decay over time, but the time it is applied changes. This parameter does not affect Step decay. Step decay is always synchronous at generation time.

Color Demosaicing

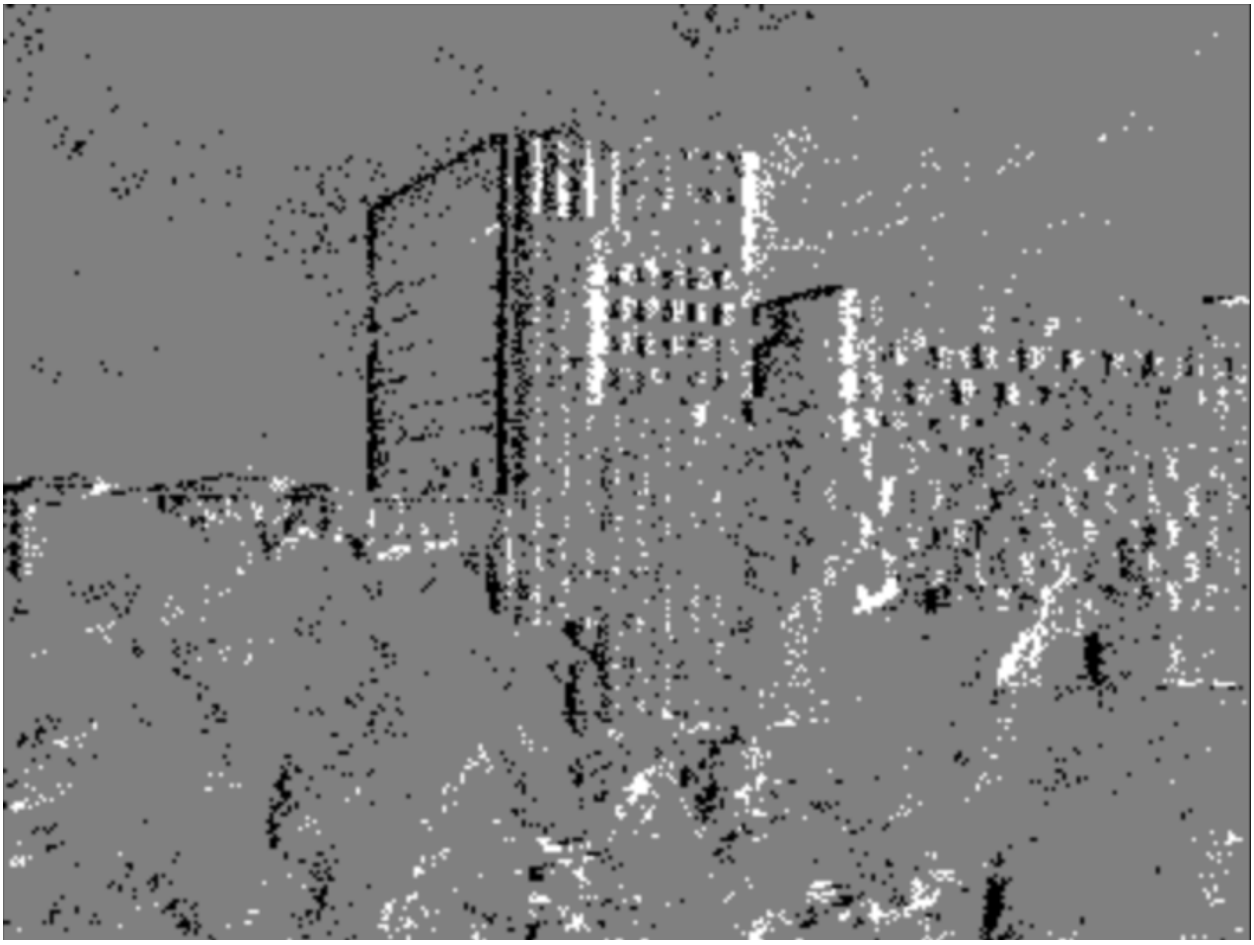
Adds a Bayer color demosaicing filter onto the generated frame. Note that this setting only makes sense when a color dvs camera is connected.

Example Configurations

High FPS Event Frames

For a use case with a very fast frame-based processing engine, one can get high-speed HDR images of the moving edges. The following settings give 1000 fps with frames that are exposed over 1 ms integration time.

Setting	Value
Accumulation time	1 (ms)
Decay function	Step
Decay param	*Ignored*
Min potential	0
Max potential	1
Neutral potential	0.5
Event contribution	0.5
Rectify polarity	No
Synchronous decay	*Ignored*

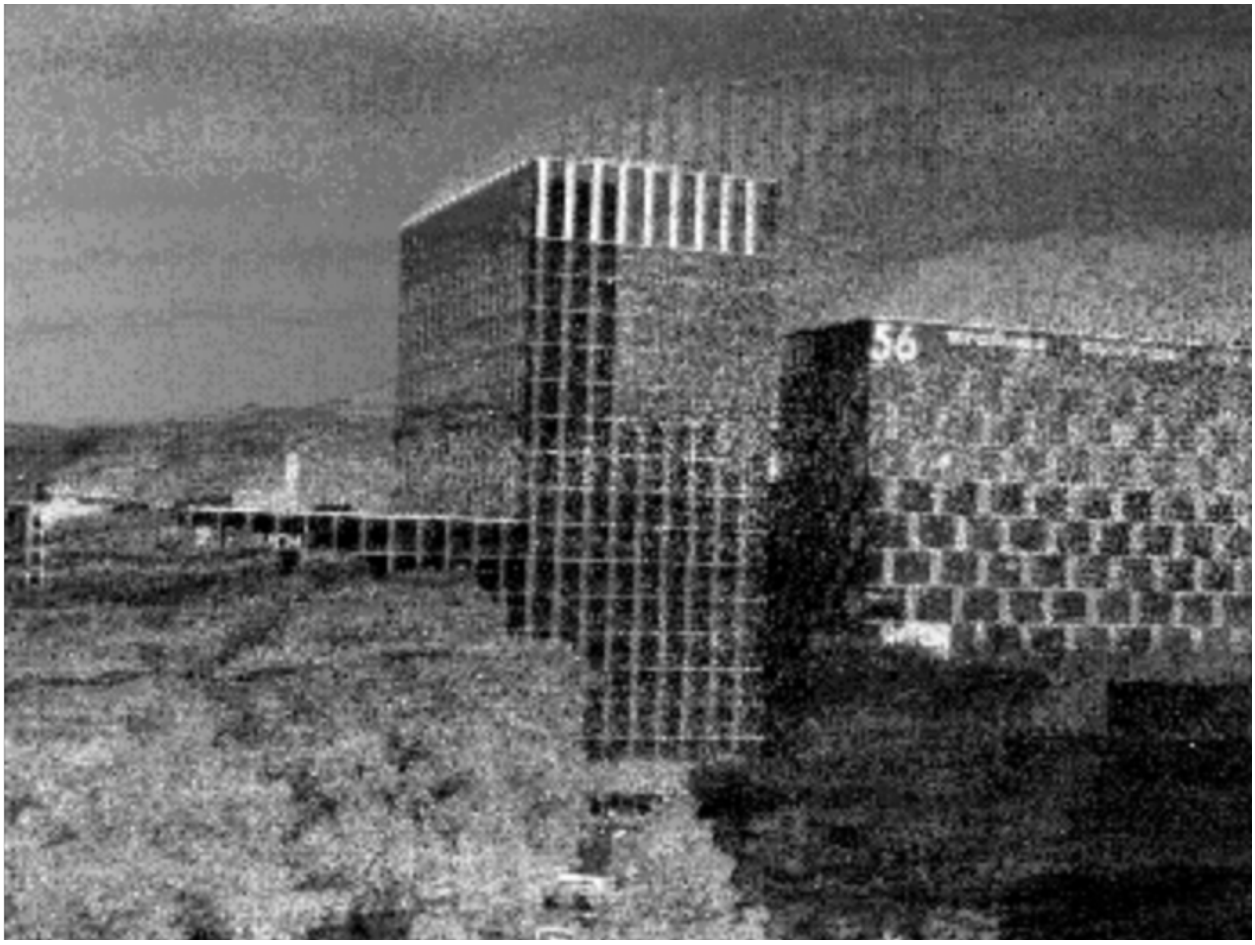


Exponentially Decayed Frame Reconstruction

The following configuration gives a good frame reconstruction if the scene has a lot of activity. The exponential decay is strong, which leads to strong shadows when a movement changes direction. The strong decay suppresses noise quite effectively.

Setting	Value
Accumulation time	33 (ms)
Decay function	Exponential
Decay param	1000000 (us)
Min potential	0
Max potential	0.3
Neutral potential	0
Event contribution	0.04
Rectify polarity	No
Synchronous decay	Yes

The fps can be arbitrarily set. In this example, *Accumulation time* is set to 33ms to achieve 30 fps. The quality is independent of the fps, if you require 1000 fps, you can set the *Accumulation time* to 1 ms.



Linear Decayed Frame Reconstruction

A reconstruction can also be achieved with a linear decay. Generally, this tends to preserve less detail but suffers less from shadows.

Setting	Value
Accumulation time	33 (ms)
Decay function	Linear
Decay param	0.000001 (intensity / us)
Min potential	0
Max potential	1
Neutral potential	0.5
Event contribution	0.15
Rectify polarity	No
Synchronous decay	Yes



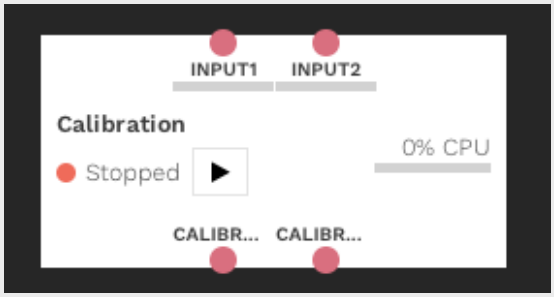
Analogous to the exponential case, one can set the fps by changing the *Accumulation time* parameter.

Further Configurations

It is advised to experiment with the accumulator settings until the desired effect is achieved.

Calibration Module

Source code[93]

Entry in *Module Selection*	Module Appearance
Dv calibration Perform lens calibration for one camera or stereo calibration for two cameras.	

[93] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/calibration

This module is used to calibrate event cameras. A fully detailed documentation of that process is available *here*.

Camera Input Modules

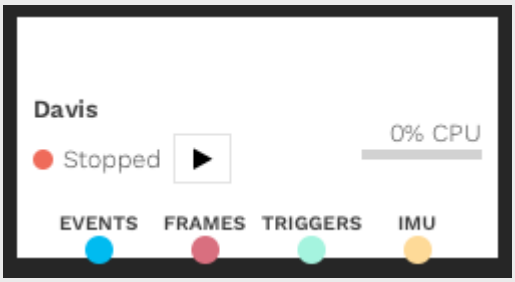
Source code[94]

These modules are used to retrieve data from our different camera products.

- We first *list and present the modules* working with our *Current Products*.
- Then, we *list and present the modules* working with custom or *Discontinued Products*.

Current Products

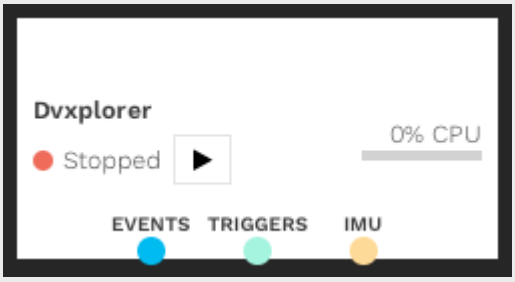
DAVIS Input Module

Entry in *Module Selection*	Module Appearance
Dv davis iniVation DAVIS camera support.	

This module retrieves data from our *DAVIS Cameras*.

It also exposes the different camera readout settings and biases that are detailed *here*.

DVXplorer Input Module

Entry in *Module Selection*	Module Appearance
Dv dvxplorer iniVation DVXplorer camera support.	

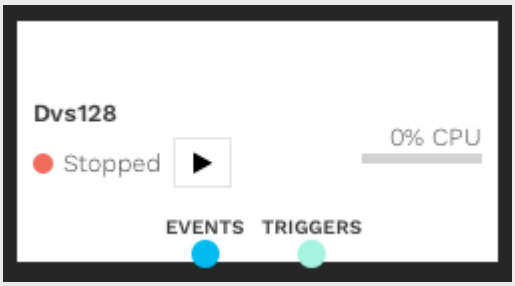
This module retrieves data from our *DVXplorer Cameras*.

It also exposes the different camera readout settings and biases that are detailed *here*.

[94] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/cameras

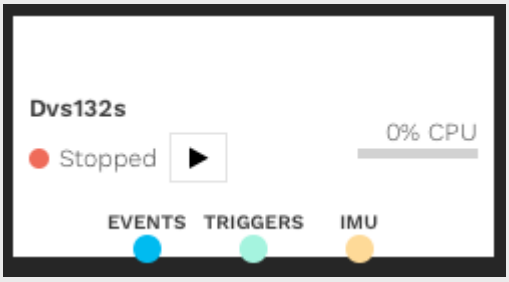
Other Products

DVS128 Input Module

Entry in *Module Selection*	Module Appearance
Dv dvs128 iniVation DVS128 camera support.	

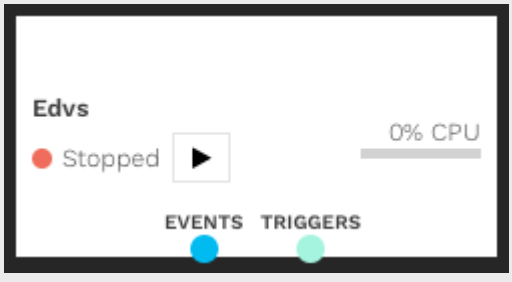
This module retrieves data from the *DVS128*.

DVS132 Input Module

Entry in *Module Selection*	Module Appearance
Dv dvs132s iniVation DVS132S camera support.	

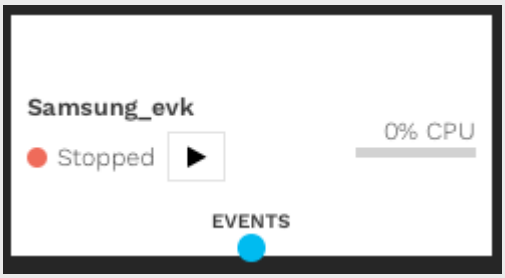
This module retrieves data from the DVS132.

eDVS

Entry in *Module Selection*	Module Appearance
Dv edvs iniVation eDVS-4337 and mini-eDVS camera support.	

This module retrieves data from the *eDVS*.

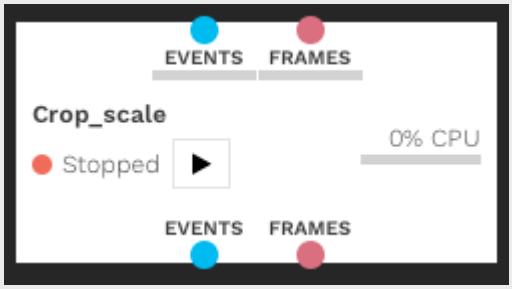
Samsung EVK

Entry in *Module Selection*	Module Appearance
Dv samsung evk Samsung EVK camera support.	

This module retrieves data from the custom Samsung EVK.

Crop and Scale Module

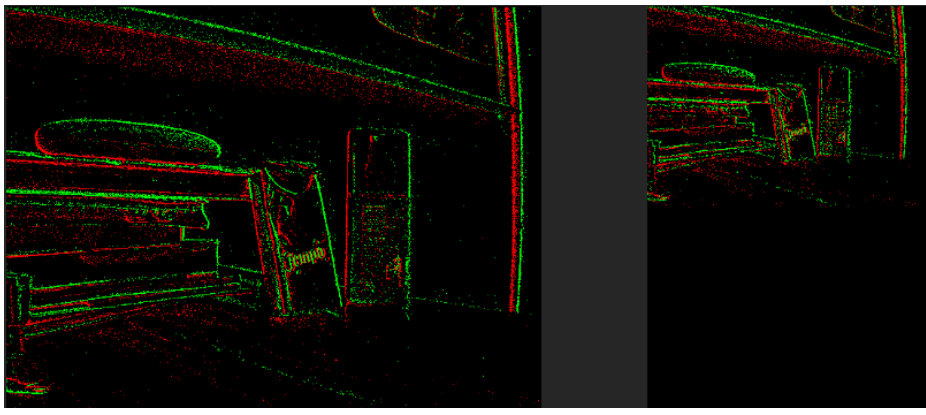
Source code[95]

Entry in *Module Selection*	Module Appearance
Dv crop scale Crop Events and Frames and resize	

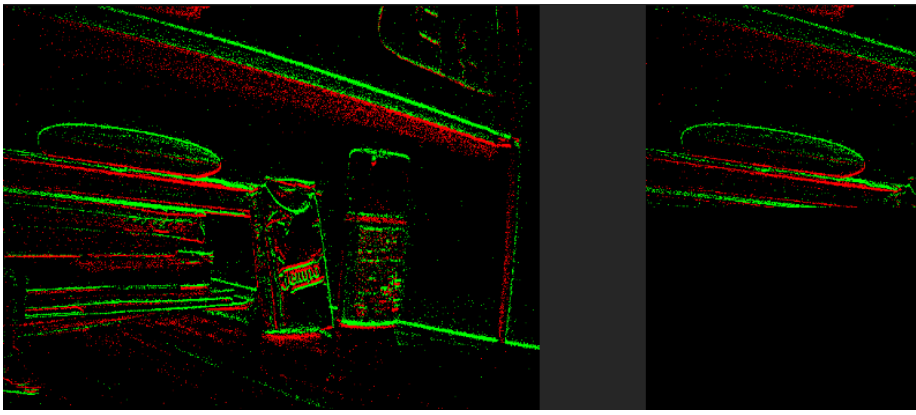
This module is used to crop or resize frames or events.

Example Usages

- Resizing Events from 640x480 to 320x240:

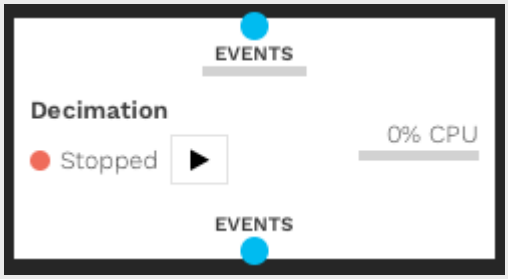


[95] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/crop_scale

- Cropping Events from 640x480 to 320x240: 

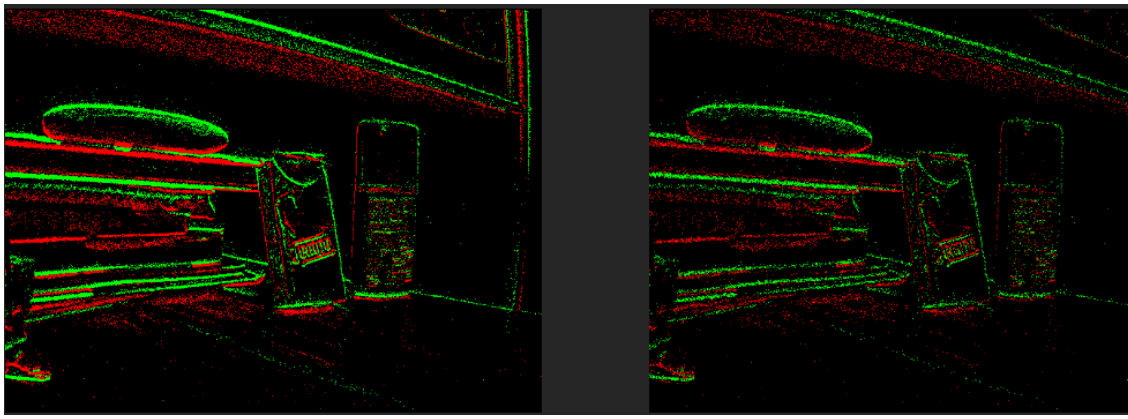
Decimation Module

Source code[96]

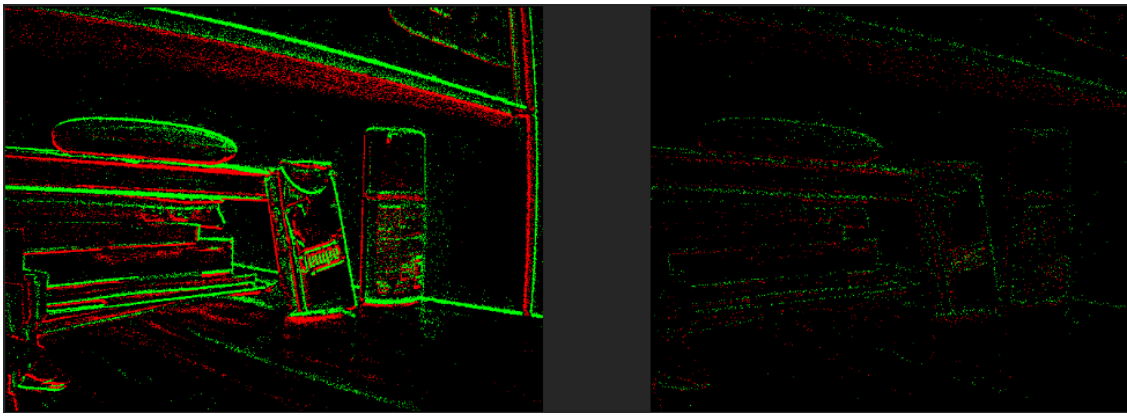
Entry in *Module Selection*	Module Appearance
Dv decimation Reduce the quantity of events to a given random percentage of the input.	

This module randomly keeps (randomly drops) some of the provided events.

Example Usages

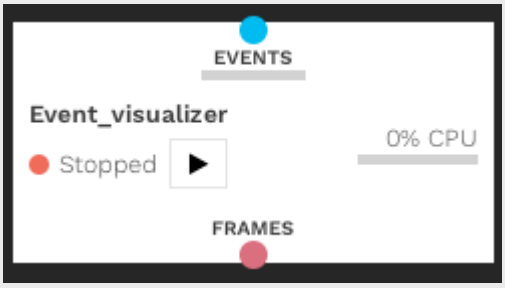
- Keeping 50% of events: 

[96] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/decimation

- Keeping 10% of events: 

Event Visualizer Module

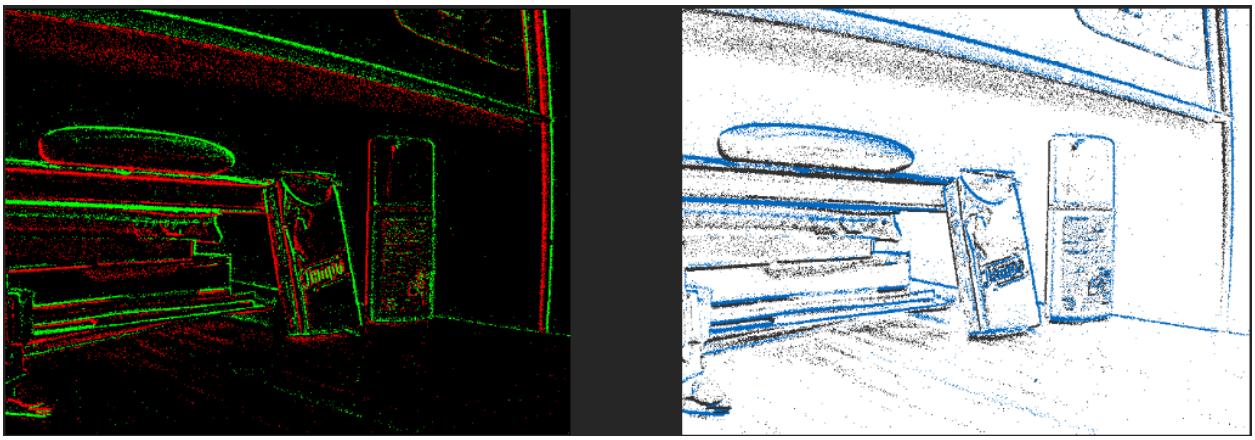
Source code[97]

Entry in *Module Selection*	Module Appearance
Dv event visualizer Simple event visualizer.	

This module visualizes events as a frame by drawing the raw events at 30 FPS.

Example Usage

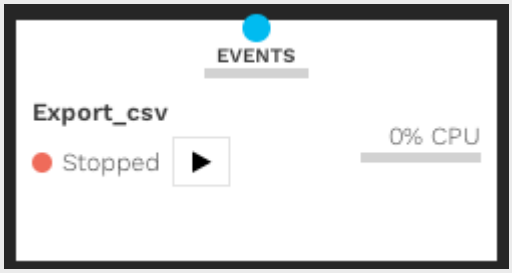
Create a frame with specific color coding (here, white background, gray negative events and blue positive events):



[97] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/event_visualizer

Export to CSV Module

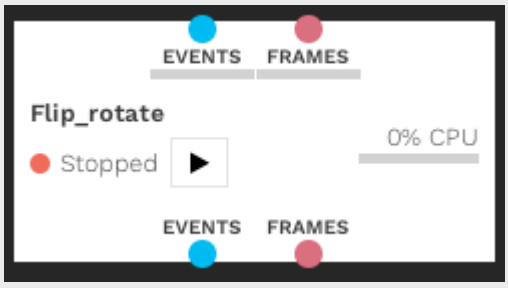
Source code[98]

Entry in *Module Selection*	Module Appearance
Dv export csv Writing event in Comma-Separated-Value format.	

This module saves events to a `.csv` file (timestamp, x, y, polarity).

Flip and Rotate Module

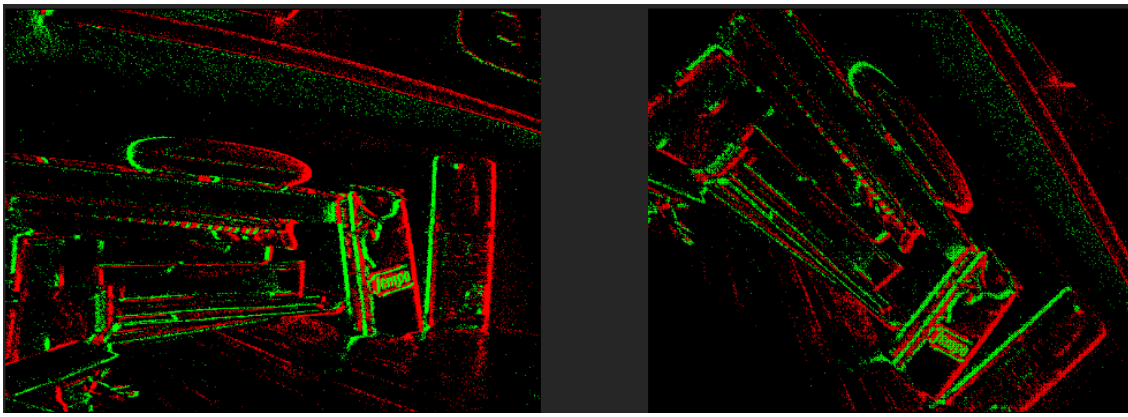
Source code[99]

Entry in *Module Selection*	Module Appearance
Dv flip rotate Flip and rotate events and frames horizontally, vertically or by setting a degree of rotation.	

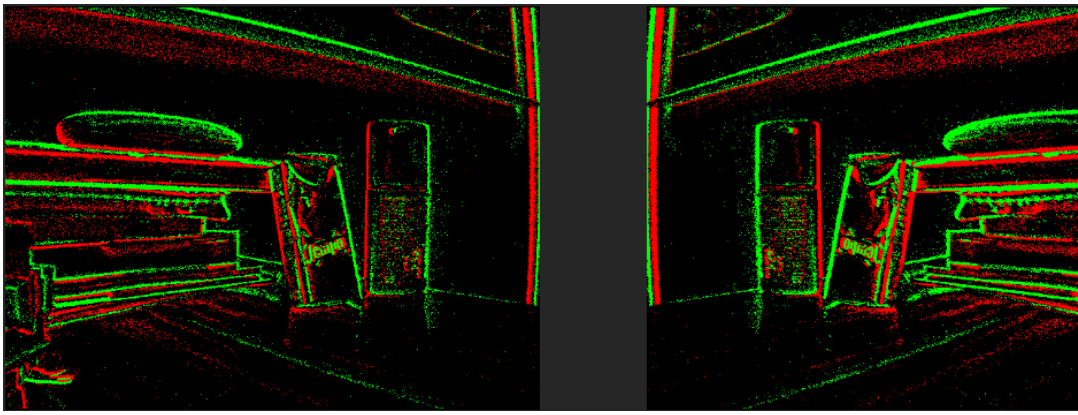
This module flips (vertically, horizontally) and/or rotates (-180° to 180°) events and/or frames.

Example Usages

- Rotating Events by 45°:

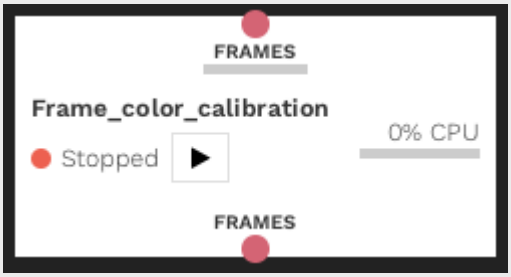


[98] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/export_csv
[99] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/flip_rotate

- Flipping events horizontally: 

Frame Color Calibration Module

Source code[100]

Module Selection	Module Appearance
Dv frame color calibration Color calibration module.	

This module performs color calibration and color correction for RGB frames.

Usage

Introduction

The idea with color calibration is to use a color checker (ex MacBeth Color Checker or Spyder Checkr) to find a mapping between uncalibrated and true color space. In this implementation, the color checker is detected, and the uncalibrated colors of the checker are stored. Since we know the calibrated colors of the checker, we can find a mapping between the uncalibrated and true known colors. This mapping can then correct the colors of images from the same sensor and lighting conditions. Below is an example where our DAVIS camera is calibrated.

[100] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/frame_color_calibration



Original frame to the left and color corrected (post calibration) to the right.

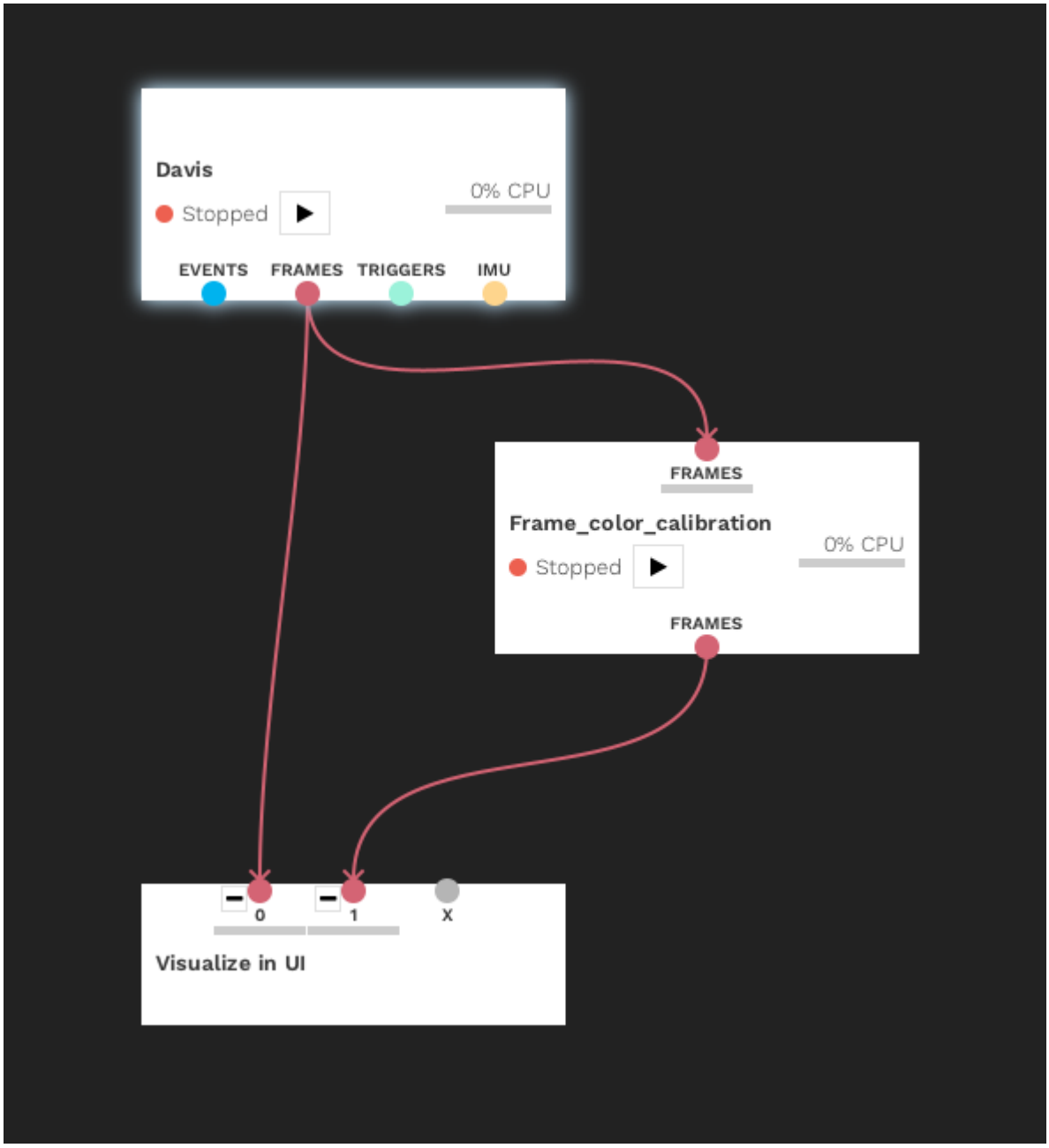
Perform Calibration

Before continuing, ensure you have a supported color checker (either MacBeth ColorChecker[101] or Spyder Checkr[102]) at hand. It is also important to have very good lighting. The color checker should be well-lit and the lighting should be uniform.

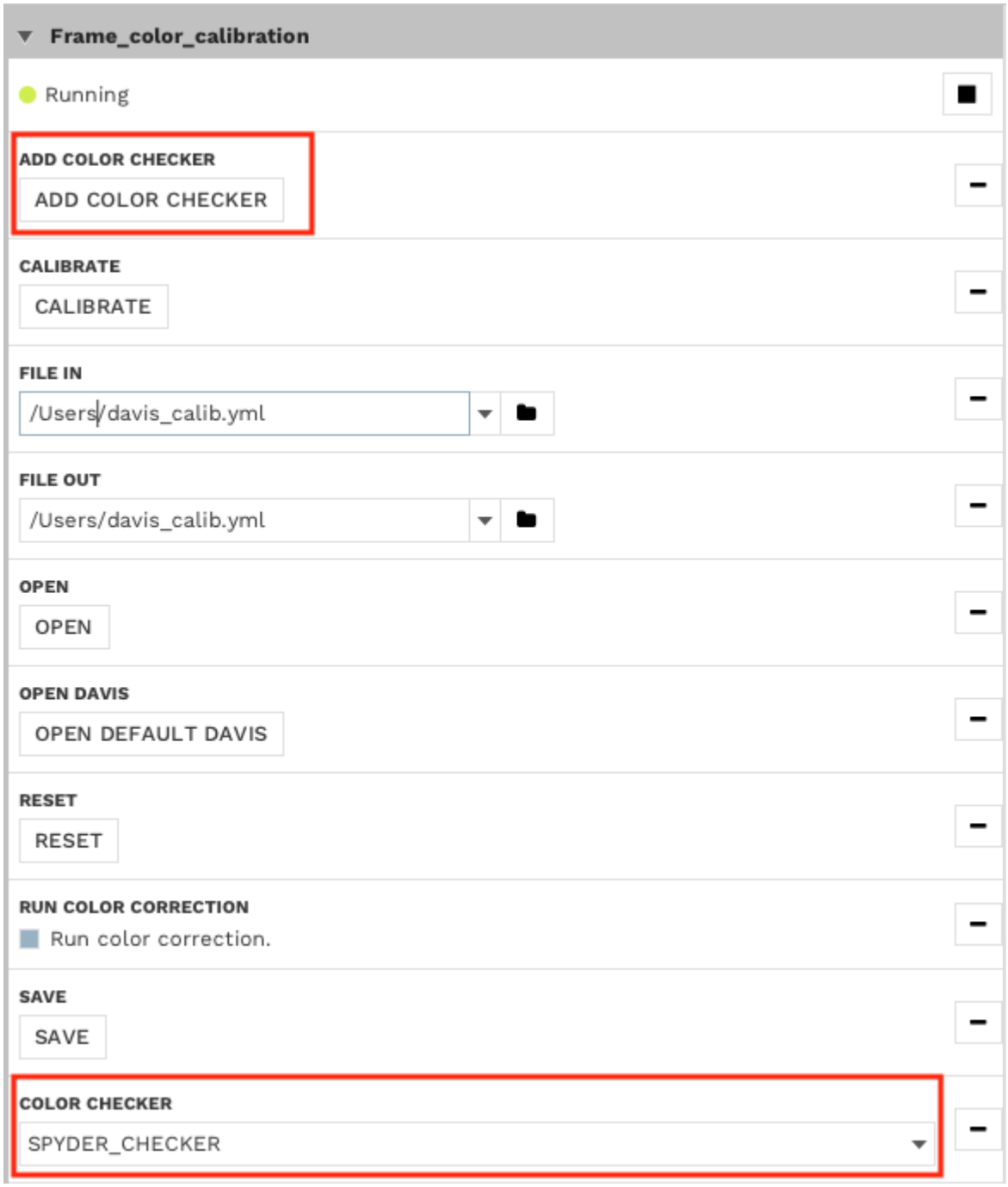
To perform color calibration using this module, follow the steps below:

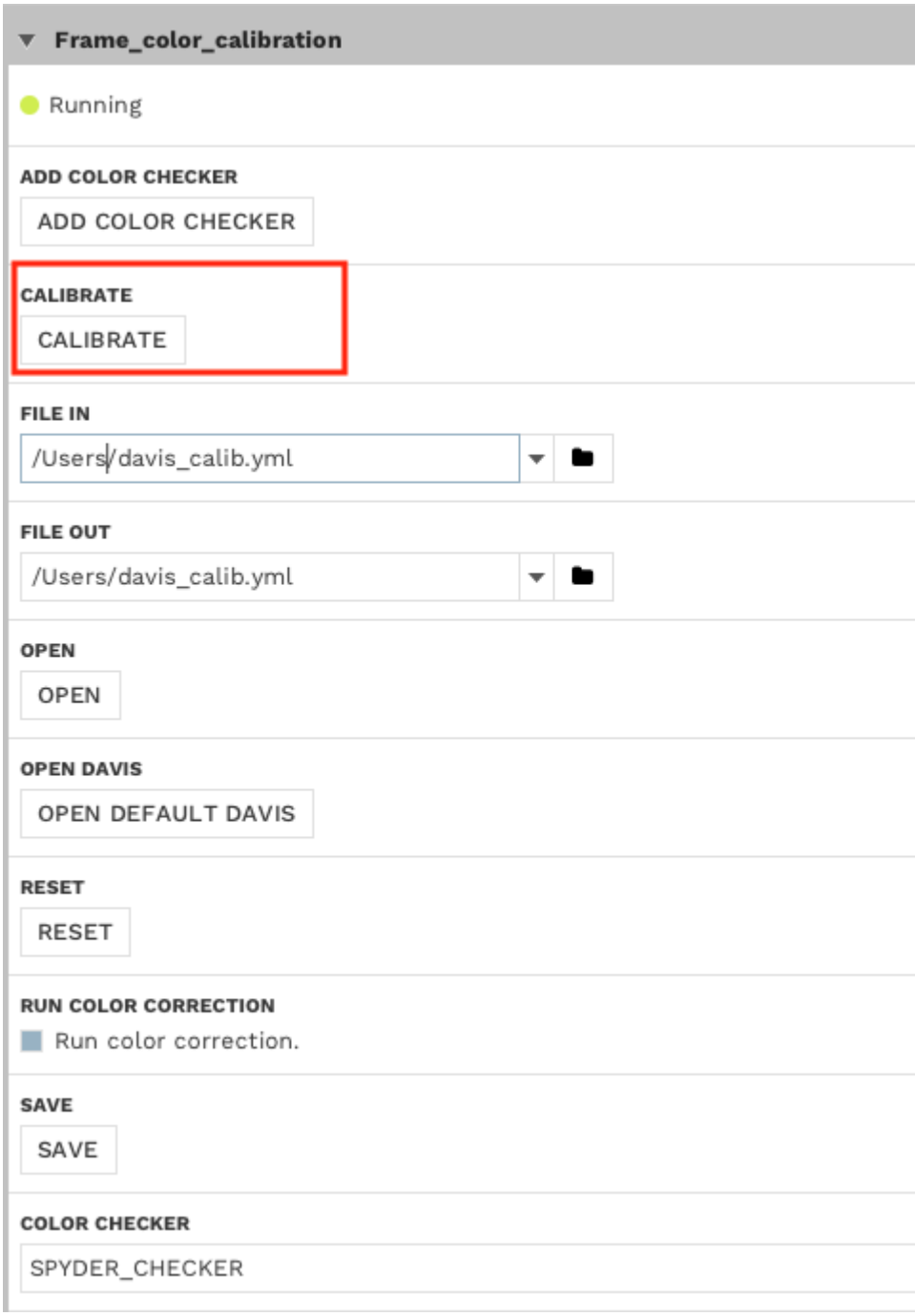
1. Set up a structure as below with RGB frames going in and visualize both the input and the output frame.

[101] https://upload.wikimedia.org/wikipedia/commons/4/4f/Color_Checker.pdf
[102] https://www.datacolor.com/spyder/products/spyder-checkr/

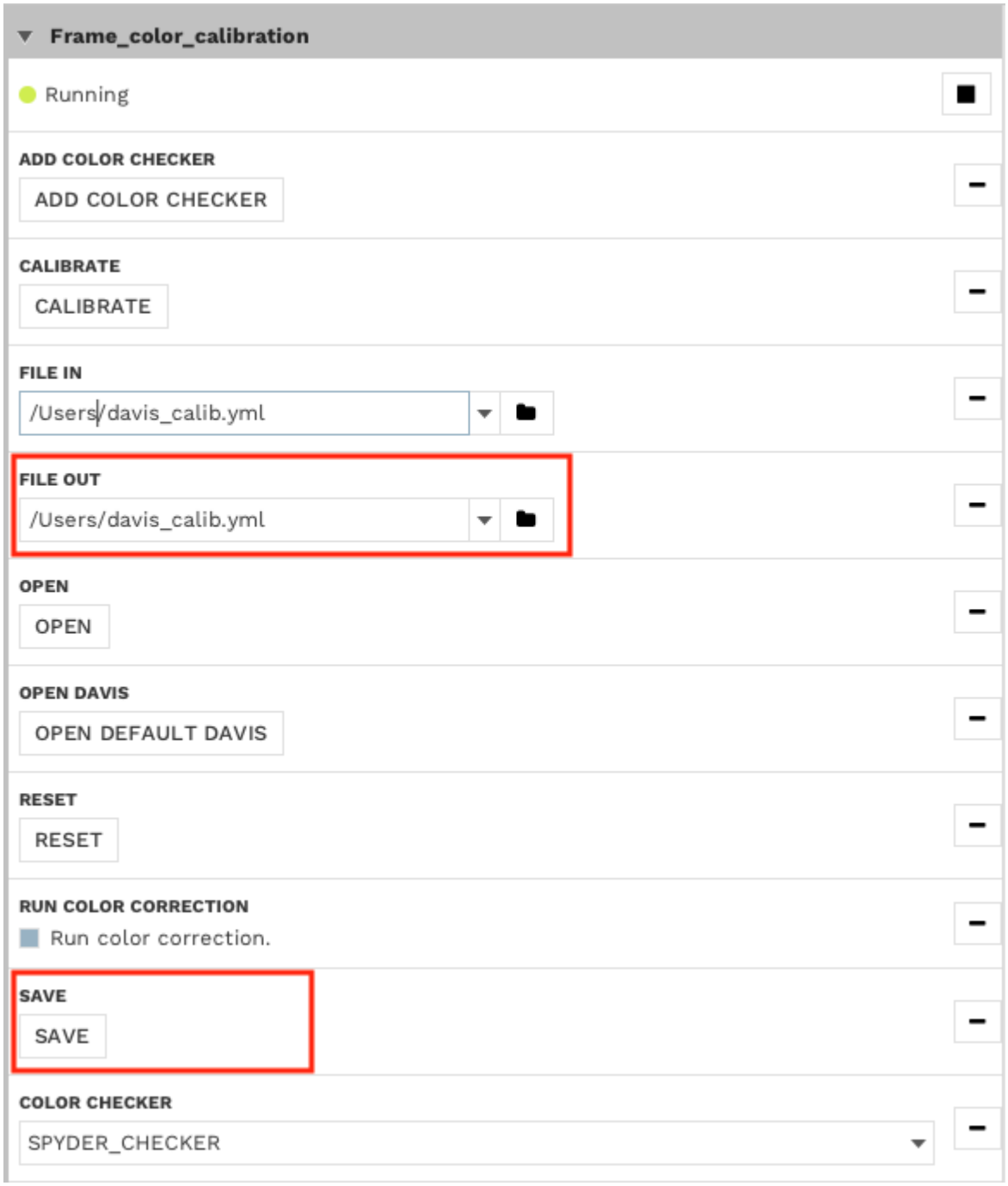


2. Start the input module and the calibration module.
3. Make sure to have the color checker well in frame and that the checker is well-lit.
4. Choose the **COLOR CHECKER** type in the module settings and press **ADD COLOR CHECKER**. To improve calibration, add more color checkers by pressing **ADD COLOR CHECKER** multiple times.


Frame_color_calibration
Running
ADD COLOR CHECKER
ADD COLOR CHECKER
CALIBRATE
CALIBRATE
FILE IN
/Users/davis_calib.yml
FILE OUT
/Users/davis_calib.yml
OPEN
OPEN
OPEN DAVIS
OPEN DEFAULT DAVIS
RESET
RESET
RUN COLOR CORRECTION
Run color correction.
SAVE
SAVE
COLOR CHECKER
SPYDER_CHECKER



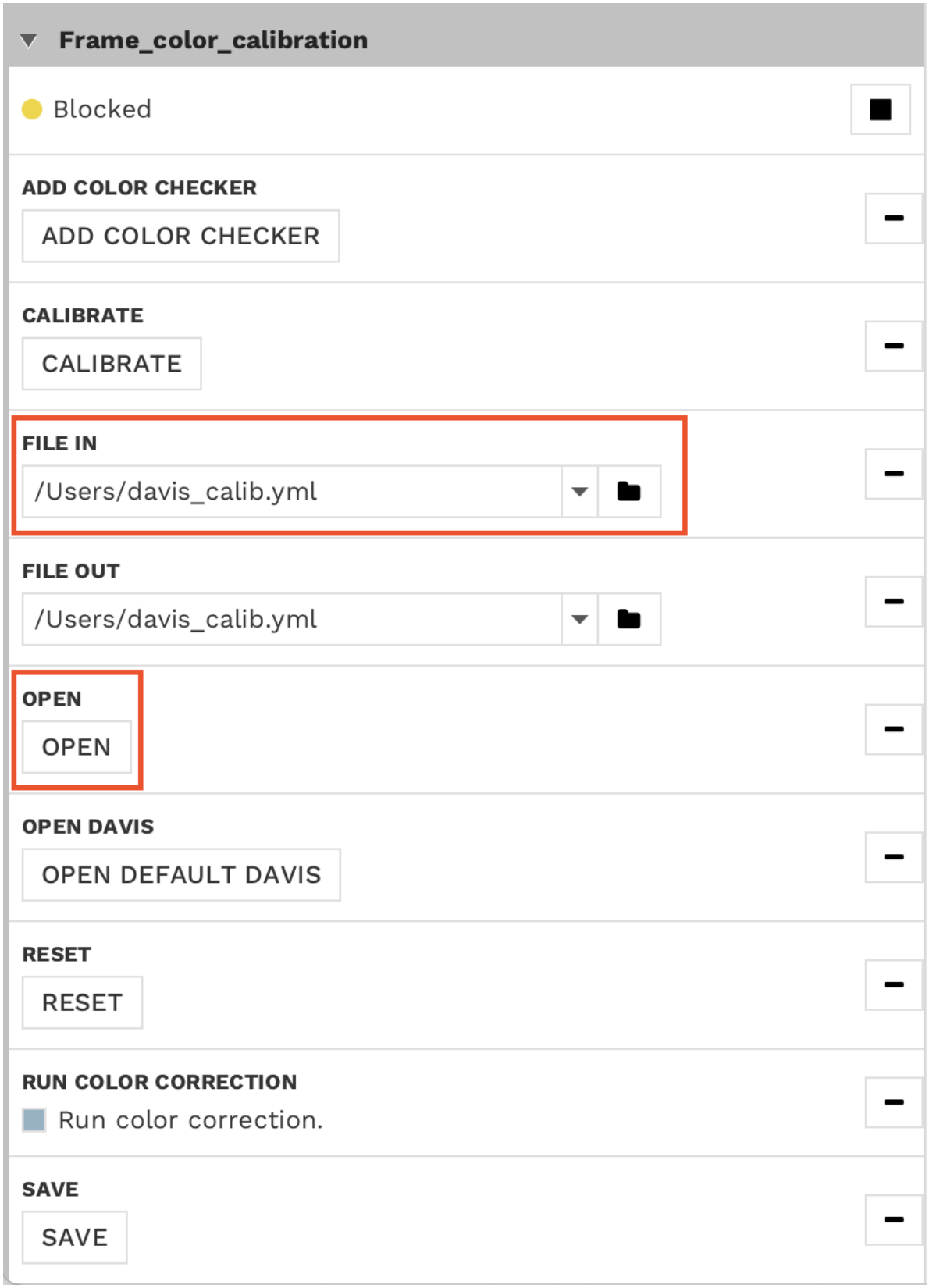
5. Press **CALIBRATE** to start calibration.
6. The color correction will start automatically after calibration is done. Evaluate the result visually. If needed, redo color checker acquisition and calibration by clicking **RESET**.
7. Save calibration by filling in a desired output path (in **FILE OUT** field) and pressing **SAVE**.



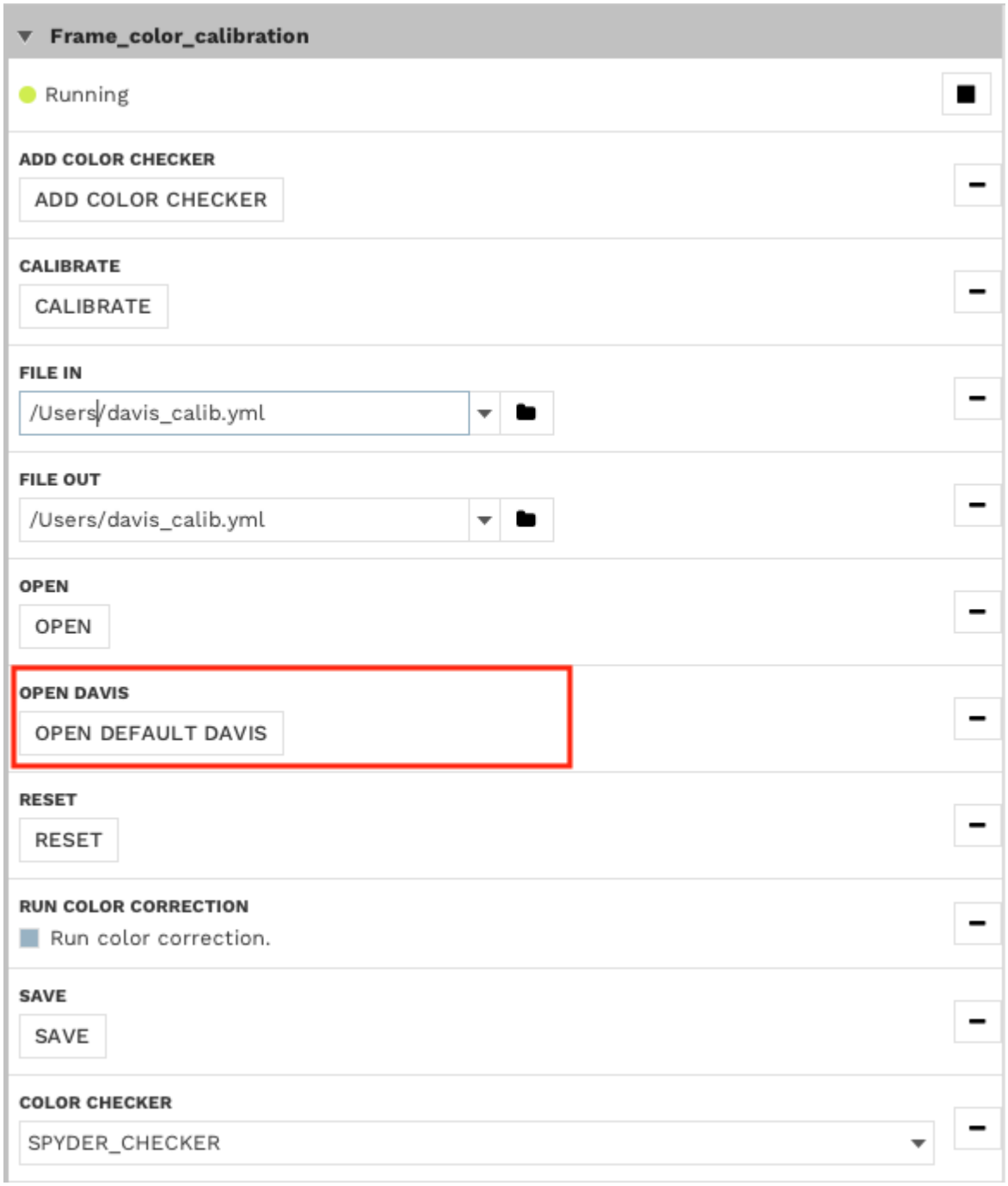
Run Color Correction

To run color correction, first do calibration, open a saved calibration or use the default Davis calibration.

- See above how to perform calibration.
- To open a saved calibration, fill in the path to the calibration file in the **FILE IN** field and press **OPEN**. ‘



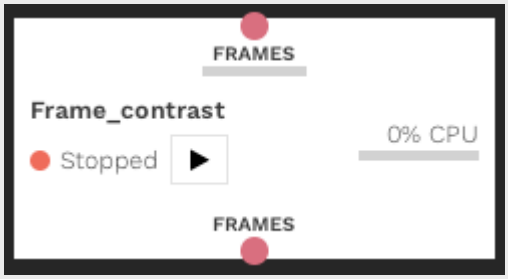
- To open the default Davis calibration, press **OPEN DEFAULT DAVIS**.



Frame Contrast Module

Source code[103]

[103] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/frame_contrast

Entry in *Module Selection*	Module Appearance
Dv frame contrast Enhance images by applying contrast enhancement algorithms.	

This module applies contrast enhancement techniques to images.

Example Usages

It exposes 3 different techniques:

- Normalization[104]



- Histogram Equalization[105]



[104] https://en.wikipedia.org/wiki/Normalization_(image_processing)
[105] https://en.wikipedia.org/wiki/Histogram_equalization

- CLAHE Algorithm[106].



Frame Histogram Module

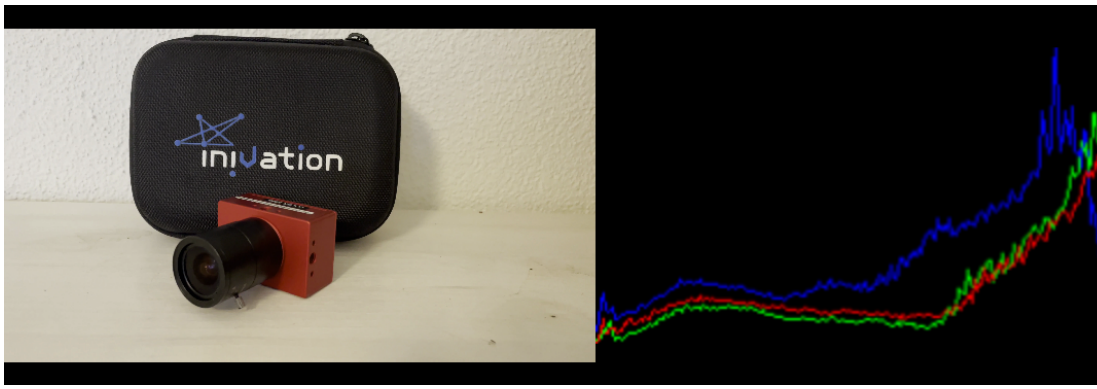
Source code[107]

Entry in *Module Selection*	Module Appearance
Dv frame histogram Display a histogram of the incoming frame	FRAMES Frame_histogram 0% CPU Stopped HISTOG...

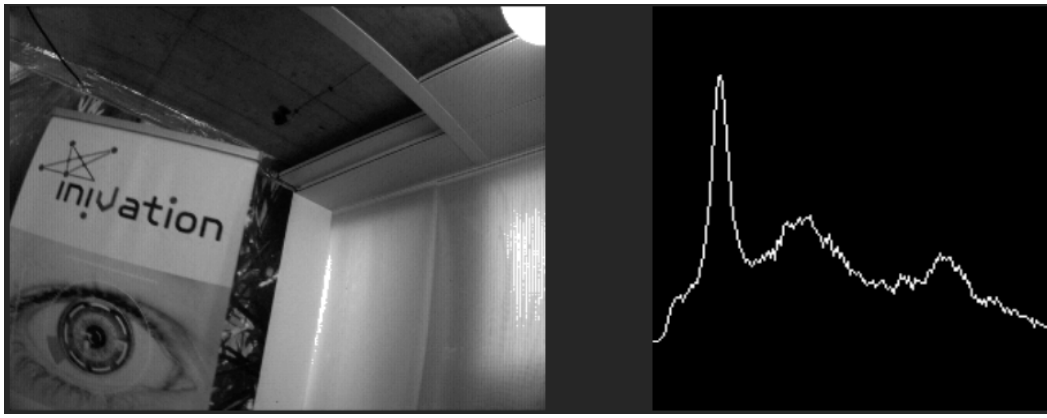
This module visualizes image histogram[108] to show the tone distribution of provided frames.

Example Usages

- Histogram of RGB Frame:



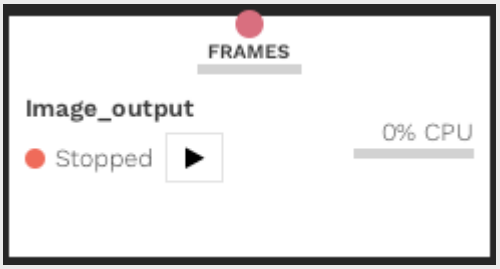
[106] https://en.wikipedia.org/wiki/Adaptive_histogram_equalization#Contrast_Limited_AHE
[107] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/frame_histogram
[108] https://en.wikipedia.org/wiki/Image_histogram

- Histogram of Grayscale Frame: 

Frame Output Modules

Save Frames to Image Files (.png)

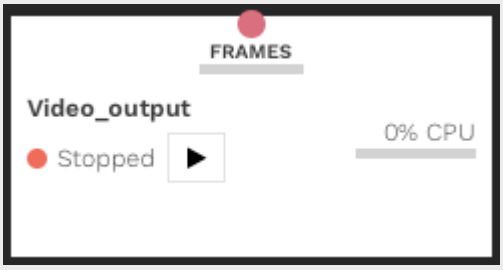
Source code[109]

Entry in *Module Selection*	Module Appearance
Dv image output Write out frames as PNG images.	

This module saves incoming images as a series of `.png` files.

Save Frames to Video File (.mkv)

Source code[110]

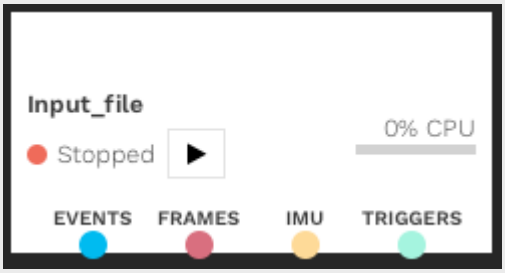
Entry in *Module Selection*	Module Appearance
Dv video output Make a video file out of frames.	

This module saves incoming images as a single video in a `.mkv` file.

[109] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/image_output
[110] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/video_output

Input File Module

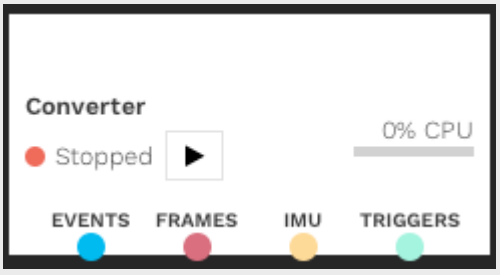
Source code[111]

Entry in *Module Selection*	Module Appearance
Dv input file Read AEDAT 4.0 data from a file.	

This module reads *AEDAT4 files*.

Old AEDAT2/AEDAT3 Files

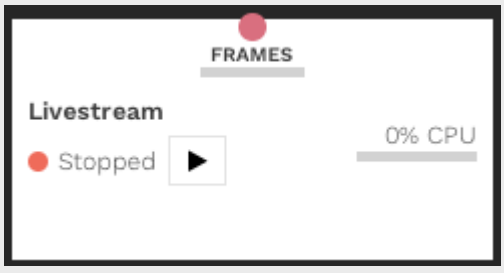
Source code[112]

Entry in *Module Selection*	Module Appearance
Dv converter Reads aedat 2 / 3 files and outputs them, so that they can be re-written to a newer format.	

This module reads *AEDAT2, AEDAT3 files* and converts the data to *AEDAT4 format* that can be processed in DV. Note that the AEDAT2 and AEDAT 3 file formats are outdated and replaced with AEDAT4 file format.

Live Stream Module

Source code[113]

Entry in *Module Selection*	Module Appearance
Dv livestream Stream live video via FFMPEG.	

[111] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/input
[112] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/convert
[113] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/livestream

This module streams provided frames as a video feed via FFMPEG[114].

Noise Filter Modules

Noise Filters are usually employed to clean the input from DVS cameras; due to the architecture of the sensors, they are sensitive to Background Activity noise produced by temporal noise and junction leakage currents. Background activity increases with low light or higher sensitivity, so a filter is often needed for applications in such conditions. In some cases, a noise filter can be useful for eliminating real events due to minor light changes and keeping a better separation between the moving subject and the background.

Use cases:

- low light conditions
- high sensitivity biases
- dark background

Filters

Dvsnoise Filter

Source code[115]

Entry in *Module Selection*	Module Appearance
Dv dvsnoisefilter Filters out noise from DVS change (polarity) events.	EVENTS Dvsnoisefilter Stopped ▶ 0% CPU EVENTS

The Dvsnoise filter is a standard spatiotemporal filter that stores a map of event timestamps. The filter also provides a hot-pixel filter with learning to filter "broken" (always-active) pixels in the sensor, as well as a refractory period filter, which limits the rate at which single pixels can consecutively generate events.

This filter provides a fixed mask size of 8 neighboring events and several advanced configurable options.

[114] https://www.ffmpeg.org/
[115] https://gitlab.com/inivation/dv/dv-runtime/-/tree/master/modules/dvsnoisefilter

Dvsnoise Filter Parameters

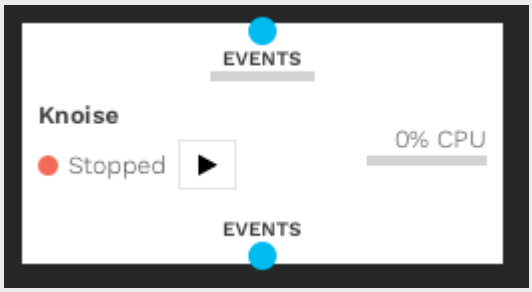
Parameter	Description
`backgroundActivityTime`	Maximum time difference in µs for events to be considered correlated and not be filtered out
`backgroundActivityEnable`	Enable the background activity filter
`refractoryPeriodTime`	Minimum time between consecutive events to not be filtered out
`refractoryPeriodEnable`	Enable the refractory period filter
`hotPixelLearn`	Learn the position of current hot (abnormally active) pixels
`hotPixelEnable`	Enable the hot pixel filter

Dvsnoise Filter Recommended Use Cases

- Generic filter
- Static camera

Knoise Filter

Source code[116]

Entry in *Module Selection*	Module Appearance
	

The Knoise filter is based on the paper "O(N)-Space Spatiotemporal Filter for Reducing Noise in Neuromorphic Vision Sensors[117]"; it is a spatiotemporal filter with memory optimizations. Memory usage and accesses are greatly reduced by only storing one timestamp per row and column.

This filter provides only 2 parameters, the time interval to look up the spatiotemporal neighbors (deltaT) and the number of supporting pixels (supporters). As the paper shows, this filter is very memory efficient but is prone to false negatives, meaning that real events can be filtered out because of missing past information. Currently, the default parameter for deltaT is 1ms.

Knoise Parameters

Parameter	Description
`deltaT`	Time delta for the filter in µs
`supporters`	Number of supporting pixels

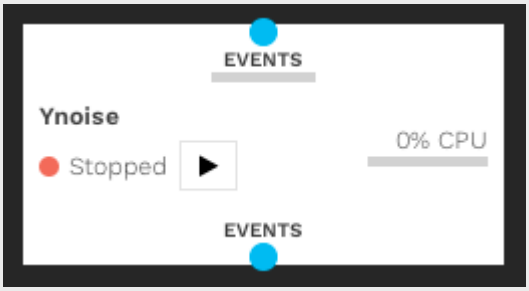
[116] https://gitlab.com/inivation/dv/dv-runtime/-/tree/master/modules/knoise
[117] http://kastner.ucsd.edu/wp-content/uploads/2013/08/admin/tetc18-neuromorphic_noise.pdf

Knoise Recommended Use Cases

- Moving camera
- Embedded systems

Ynoise Filter

Source code[118]

Entry in *Module Selection*	Module Appearance
Dv ynoise Noise filter using the Yang algorithm.	

The Ynoise filter is based on the paper "Event Density Based Denoising Method for Dynamic Vision Sensor[119]", that describes a variant of the standard spatiotemporal filter.

Currently, we have only implemented what they call `coarse filtering` to achieve better performance. The complete algorithm adds one further step called `fine filtering`, used to remove flickering noise. We also use the L∞ norm because it is easier to compute with a representation of the density matrix in an array structure.

Ynoise Parameters

Parameter	Description
`deltaT`	Time delta for the filter in µs
`lParam`	Value L of LxL matrix for density matrix
`threshold`	Threshold value for density pixels

Ynoise Recommended Use Cases

- Generic filter
- Static camera
- Image reconstruction

Comparison

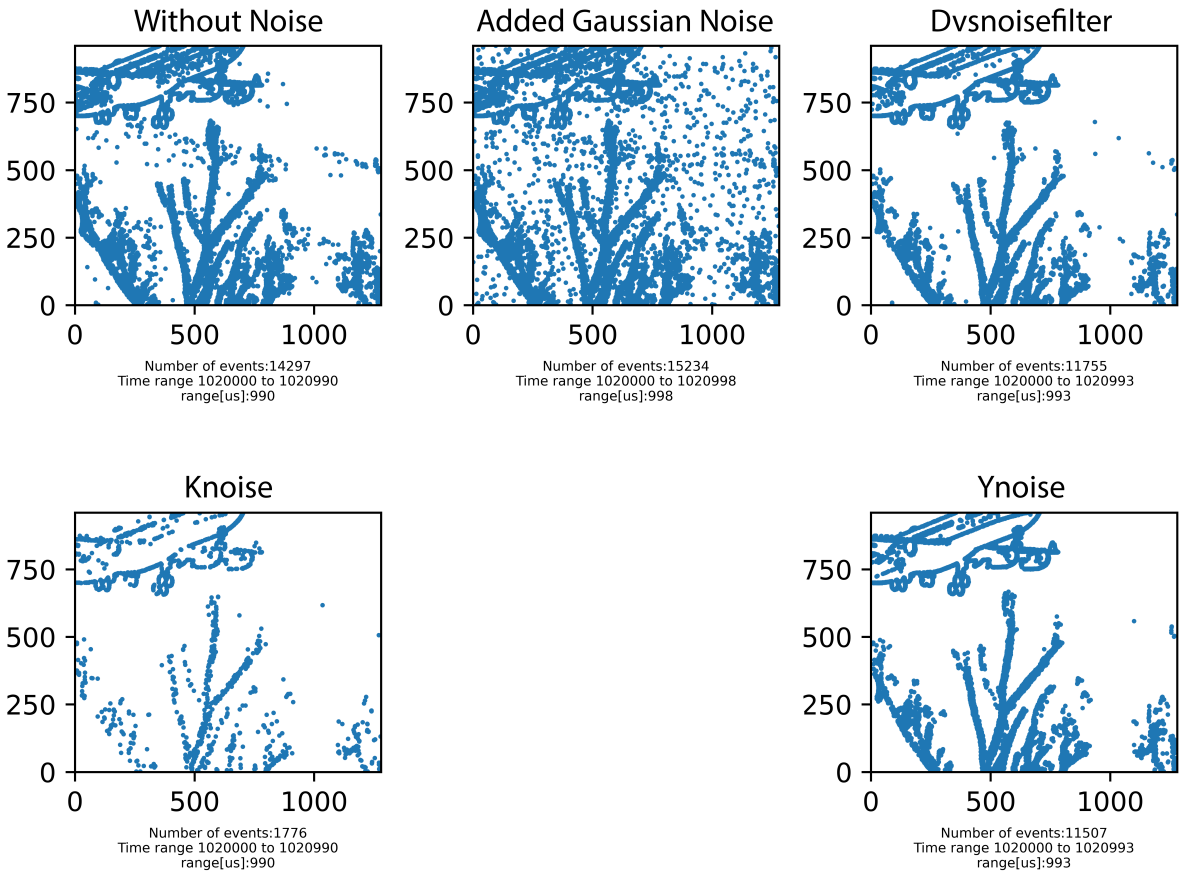
Comparing noise filters can be tricky because the performance and quality of the filter depend heavily on the parameters. Generally, better quality means worse performance. For the following comparison, we used the most balanced settings, which are currently the default settings in DV-Runtime.

[118] https://gitlab.com/inivation/dv/dv-runtime/-/tree/master/modules/ynoise
[119] https://www.mdpi.com/2076-3417/10/6/2024

We artificially generated a sequence of events without any noise, then added Gaussian noise to it to generate meaningful and precise metrics on noise filtering, signal filtering and so on. While not a perfect setup, it gives indicative results and is deterministic and reproducible.

Visual Comparison

As shown in the image below, we can see that the Dvsnoise filter keeps the most real events after filtering. The Ynoise filter loses some real events, but most of the shapes of the plane and tree are intact. Instead, Knoise is already corrupting the shape of the plane due to the massive filtering going on.



Note: data100k is an artificially generated event file without noise, data_noise100k is that file with added Gaussian noise, and the other images are the output of the discussed filters.

Signal to Noise Ratio

Below, we present a table with the different signal-to-noise ratios, calculated with default settings (most balanced) with our sample event files. As we can see, the two most balanced filters are the Dvsnoise filter and the Ynoise filter. They are slower than the Knoise filter but have a better SR (Signal Ratio). The SNR (Signal to Noise Ratio) is higher in the Knoise filter because most of the events are eliminated (noise included), making the SNR higher and the SR lower, though it still is the fastest of the tested filters due to its memory trade-offs.

The formulas for calculating the SR, NR and SNR are:

$$SR = \frac{E^{f}_{signal}}{E^{o}_{signal}}, \quad NR = \frac{E^{f}_{noise}}{E^{o}_{noise}}, \quad SNR^{q} = \frac{E^{q}_{si}}{E^{q}_{n}}$$

Where: Ef: stand for number of events after filtering Eo: stands for number of events before filtering Eq: can be f or o signal: stands for original video noise: stands for video with noise

Parameters (same for both event files)

- **Dvsnoise filter**
  - `hotPixelEnable`: false
  - `backgroundActivityEnable`: true
  - `backgroundActivityTwoLevels`: false
  - `backgroundActivityCheckPolarity`: false
  - `backgroundActivitySupportMin`: 1
  - `backgroundActivitySupportMax`: 8
  - `backgroundActivityTime`: 2000
  - `refractoryPeriodEnable`: true
  - `refractoryPeriodTime`: 100
- **Knoise**
  - `deltaT`: 1000
  - `supporters`: 1
- **Ynoise**
  - `deltaT`: 10000
  - `lParam`: 3
  - `threshold`: 2

Results

- **Event File 1 (duration 5s, 1280x960)**

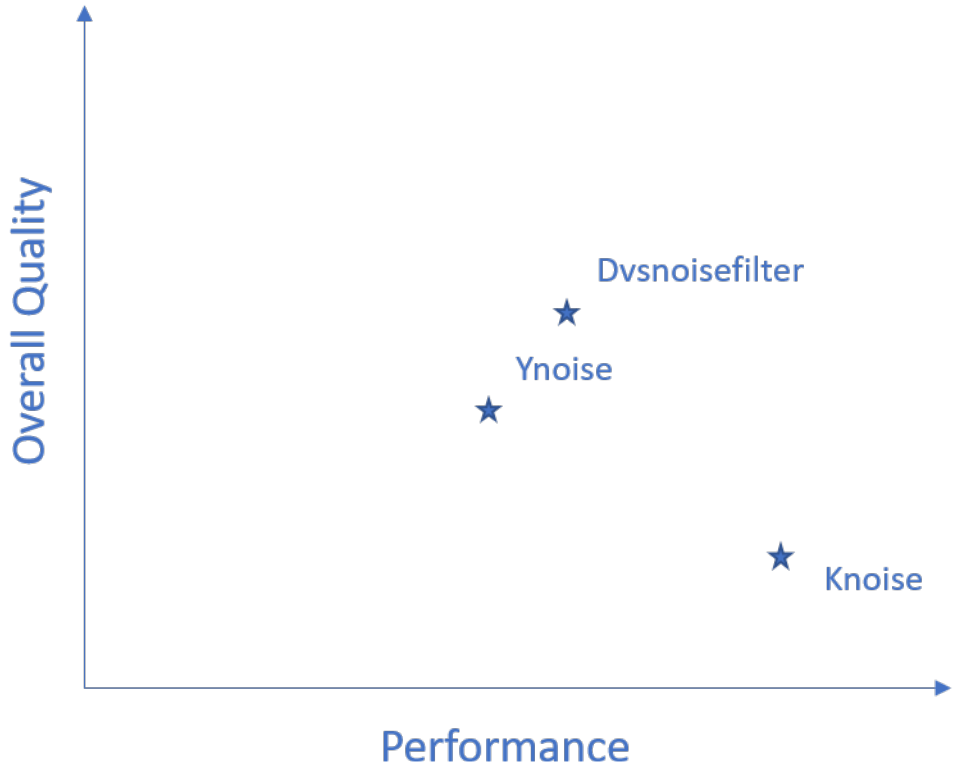
Filter	SR ↑	NR ↓	SNR ↑	CPU time (kernel time + user time) ↓
Dvsnoise	0.9	0.08	208.06	7.92s
Knoise	0.19	0.0	791.9	4.35s
Ynoise	0.58	0.03	388.69	5.25s

- **Event File 2 (duration 5s, 640x480)**

Filter	SR ↑	NR ↓	SNR ↑	CPU time (kernel time + user time) ↓
Dvsnoise	0.87	0.22	38.06	4.70s
Knoise	0.15	0.01	101.56	2.02s
Ynoise	0.63	0.07	86.93	4.92s

Note: CPU time is relative to the system in use, so it is only to be used as an indicator for comparison between filters.

Quality and Performance Comparison Chart



We recommend for standard usage (with static cameras or without significant scene changes) the Dvsnoise filter and the Ynoise filters. For dynamic camera situations or significant scene changes, we recommend the Knoise filter, as it is faster and will provide a better compromise for keeping good filter performance and real-time processing speeds.

Output File Module

Source code[120]

Entry in *Module Selection*	Module Appearance
Dv output file Write AEDAT 4.0 data to a file.	OUTPUT0 OUTPUT1 OUTPUT2 OUTPUT3 Output_file Stopped ▶ 0% CPU

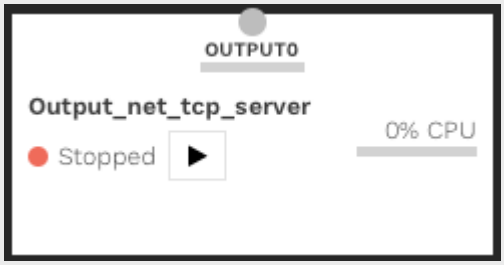
This module saves data in *AEDAT4 file format*.

[120] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/output/file.cpp

Output Network Modules

Output TCP Module

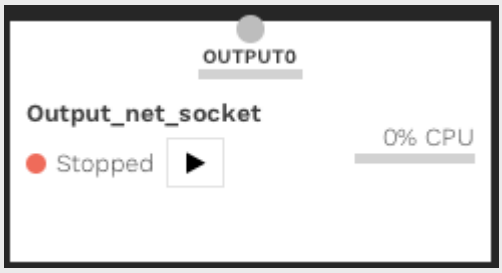
Source code[121]

Entry in *Module Selection*	Module Appearance
Dv output net tcp server Send AEDAT 4 data out via TCP to connected clients (server mode).	

This module sends *AEDAT4 data* out via TCP to connected clients (server mode).

Output Local Socket Module

Source code[122]

Entry in *Module Selection*	Module Appearance
Dv output net socket Send AEDAT 4 data out via local sockets to connected clients (server mode).	

This module sends *AEDAT4 data* out via local sockets to connected clients (server mode).

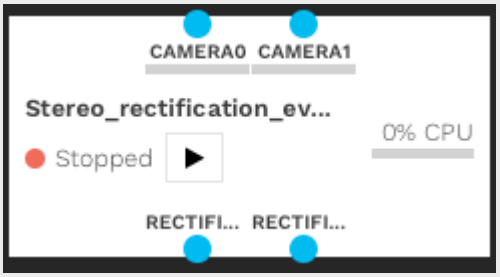
Stereo Rectification Modules

Source code[123]

These modules perform stereo rectification[124] on either events of frames.

[121] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/output/net_tcp_server.cpp
[122] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/output/net_socket.cpp
[123] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/stereo_rectification
[124] https://en.wikipedia.org/wiki/Image_rectification

Stereo Rectification on Events

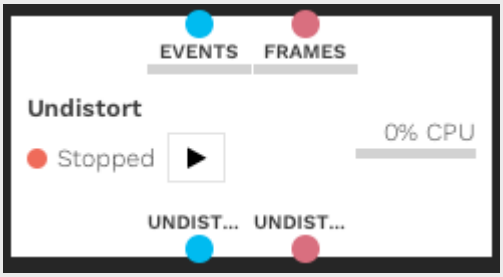
Entry in *Module Selection*	Module Appearance
Dv stereo rectification events Perform stereo rectification on events.	

Stereo Rectification on Frames

Entry in *Module Selection*	Module Appearance
Dv stereo rectification frames Perform stereo rectification on frames.	

Undistortion Module

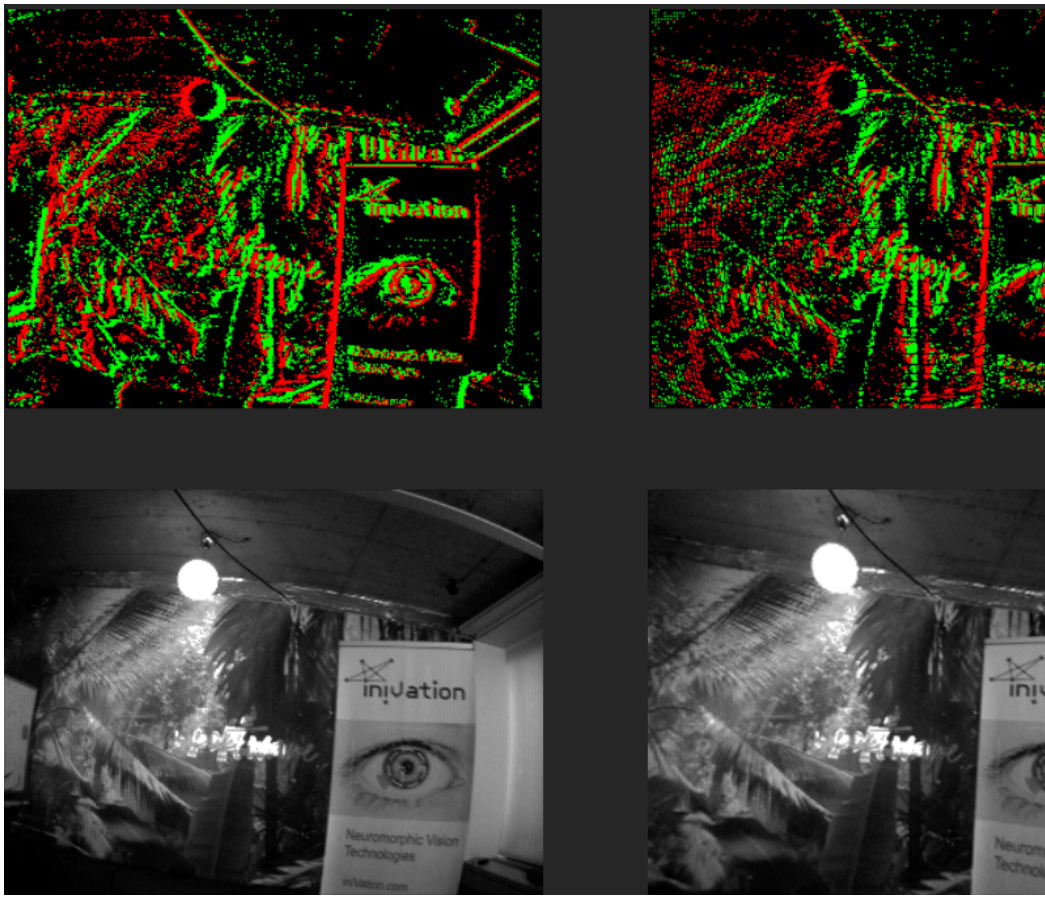
Source code[125]

Entry in *Module Selection*	Module Appearance
Dv undistort Remove distortion from lens in both frames and events.	

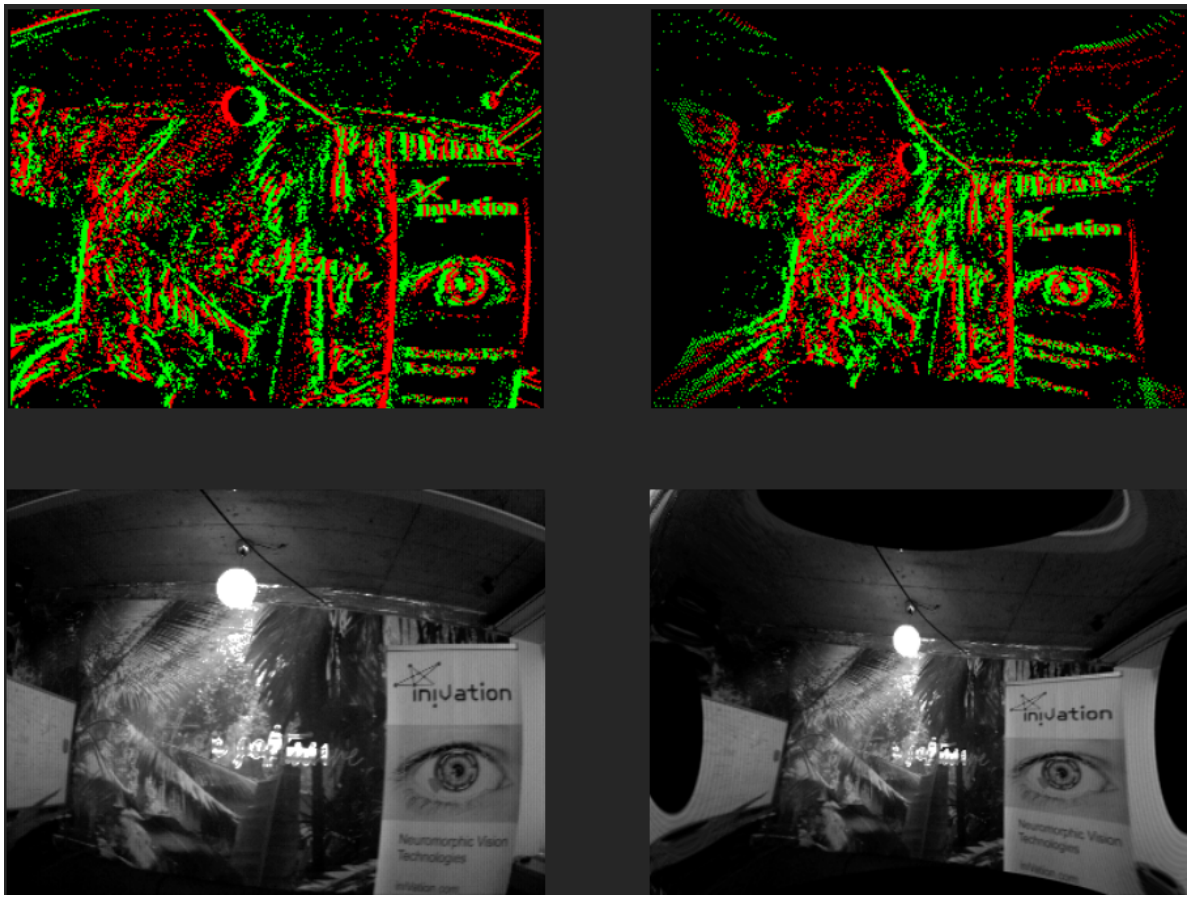
This module performs undistortion[126] on frames and/or events.

[125] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/undistort
[126] https://en.wikipedia.org/wiki/Distortion_(optics)#Software_correction

Example Usages

- Undistorting Events and Frames 

- Undistorting Events and Frames (and show all original pixels, using `FIT MORE PIXELS = 1.0`)



Visualizer Module

Source code[127]

Entry in *Module Selection*	Module Appearance
Dv visualizer Simple, lightweight frame visualizer using SDL2.	FRAMES Visualizer Stopped ▶ 0% CPU

This module visualizes frames in a separate window from the GUI.

It is also possible to create your own DV Module by using *dv-sdk*.

[127] https://gitlab.com/inivation/dv/dv-runtime/-/blob/master/modules/visualizer

5.3 DV-Runtime

Source Code[128]
Documentation[129]

`dv-runtime` is our provided Runtime System[130]. It is the environment that will host the *modules* and take care of the threads, the memory allocation and data exchange between the modules. It is better to use it implicitly via the *GUI*, but it can also be used independently.

All documentation related to the `dv-runtime` system can be found on:

https://dv-runtime.inivation.com/master/index.html#dv-runtime.

It contains details about Installation[131] and Usage[132].

5.4 DV-SDK

Source Code[133]
Documentation[134]

`dv-sdk` is the library we provide to extend the provided functionalities of the `DV` software or integrate other algorithms into it. It is used to create new *modules* that would implement new processing features or integrate existing algorithms.

All documentation related to the `dv-sdk` library can be found on:

https://dv-runtime.inivation.com/master/index.html#dv-sdk.

It contains details about Installation[135] and Usage for DV-Module creation[136].

[128] https://gitlab.com/inivation/dv/dv-runtime
[129] https://dv-runtime.inivation.com
[130] https://en.wikipedia.org/wiki/Runtime_system
[131] https://dv-runtime.inivation.com/master/runtime/installation.html
[132] https://dv-runtime.inivation.com/master/runtime/usage/index.html
[133] https://gitlab.com/inivation/dv/dv-runtime/-/tree/master/include/dv-sdk
[134] https://dv-runtime.inivation.com
[135] https://dv-runtime.inivation.com/master/sdk/installation.html
[136] https://dv-runtime.inivation.com/master/sdk/dv-modules/index.html

CHAPTER

SIX

DV-PROCESSING

Source Code[137]
Documentation[138]

DV-Processing is a library for generic processing algorithms for event cameras available both in C++ and Python.

It allows users to:

- Directly control and access data from our cameras.
- Read and Write data from/to files.
- Process event data (using convenient data structures). For example:
  - Filter events
  - Reconstruct frames
- Use state-of-the-art vision algorithms
  - Camera Geometry (Calibration)
  - Feature Detection and Tracking
  - Kinematics
  - Clustering (Mean-Shift)
- Use more advanced applications
  - Depth Estimation
  - Motion Compensation
  - Contrast Maximization

See the detailed and dedicated documentation at:

https://dv-processing.inivation.com.

[137] https://gitlab.com/inivation/dv/dv-processing
[138] https://dv-processing.inivation.com

CHAPTER

SEVEN

OTHER SOFTWARE

7.1 DV-Ros

The project dv-ros[139] provides iniVation event camera integration with ROS[140]. The integration is performed by implementing ROS nodes with the use of *DV-Processing*. The dv-ros also provides several event data processing algorithms wrapped in ROS nodes for immediate use for computer vision applications with event cameras and ROS.

The installation of the nodes in ROS and details of available nodes can be found in the readme of the code repository[141].

The integration is compatible with existing ROS message types from the rpg_dvs_ros[142], so any application compatible with the `rps_dvs_ros` nodes is also compatible with the `dv-ros` nodes.

7.2 Libcaer

Source Code[143]
API Documentation[144]

We provide a C/C++ library called libcaer for low-level access to our cameras.

7.3 Dv-Python - DISCONTINUED

This has been replaced with dv-processing, which has a more extensive Python API.

DV also has an integration with Python that allows control of DV via Python. Full documentation is available on the gitlab repository[145].

7.4 Older Releases of Our Software

Older versions can be found where we store all of our released software:
https://release.inivation.com/

[139] https://gitlab.com/inivation/dv/dv-ros
[140] https://www.ros.org/
[141] https://gitlab.com/inivation/dv/dv-ros
[142] https://github.com/uzh-rpg/rpg_dvs_ros
[143] https://gitlab.com/inivation/dv/libcaer
[144] https://libcaer.inivation.com
[145] https://gitlab.com/inivation/dv/dv-python

CHAPTER

EIGHT

ADVANCED USAGE

Jump into usage by following the documentation:

- Event Data is a different type of data compared to standard frame cameras, for that reason we use our own custom *file formats*.
- Finding the correct file format, took some time. We now have a fixed file format, AEDAT4, but you might encounter some old file in AEDAT2 or AEDAT3 formats, you can *convert them to AEDAT4 using our software*.
- The procedure to calibrate an Event camera is a bit different from standard frame cameras. The general idea of the procedure is explained in details in the *Camera Calibration section*.
- Our DV-GUI software is also usable with *remote dv-runtime instances*.

8.1 AEDAT File Formats

8.1.1 Introduction

Files generated by iniVation event-based processing software, such as DV and jAER, use the custom AEDAT (=Address Event DATa) format to store the event data and its timestamp information. Over the years, this format has evolved to address various needs, such as storing new data types more efficiently or having larger timestamp ranges.

- *AEDAT 4.0* is the most recent data format provided by the *DV (Dynamic Vision) framework* since June 2019.
- *AEDAT 3.1* was specified as an update to the AEDAT 3.0 format in April 2016, addressing a couple of shortcomings.
- *AEDAT 3.0* came out of research done with the cAER framework in 2015.
- *AEDAT 2.0* of the format is the most commonly found, as it was supported by jAER and used as default since 2010.
- *AEDAT 1.0* has been supported since the first jAER release in 2008.

8.1.2 Software Support

The following table summarizes what software supports which formats:

Software	AEDAT 4.0	AEDAT 4.0	AEDAT 3.1	AEDAT 3.1	AEDAT 3.0	AEDAT 3.0	AEDAT 2.0	AEDAT 2.0	AEDAT 1.0	AEDAT 1.0
	Read	Write	Read	Write	Read	Write	Read	Write	Read	Write
DV			*	*	*	*	*	*	*	*
dv-proces										
jAER			(RAW format)		(RAW format)					

- = Conversion from earlier Aedat data formats to Aedat4 is *available via DV*

8.1.3 Common Version Header

AEDAT files have a common, human-readable header format.

A header was initially optional, but with the introduction of new format revisions, it is now considered required, else it's impossible to parse the file correctly.

If it exists, it is the first content present in a file, and is made up of one or more header lines.

A header line begins with a '#' character and ends with a CRLF (Windows line ending, '\r\n').

Header lines, unless specified otherwise, are always case sensitive.

The very first header line contains information about the version of the AEDAT format employed in the rest of the file. If it is not present, version 1.0 is assumed.

The exact format of the version line is as follows:

#!AER-DATx.y\r\n

where x and y are the two version number components.

Other than the first version header line, no other header lines are required or specified in a common way across all format versions and depend completely upon the software generating them. They should only be used for informative uses and not to store critical data or metadata unless explicitly specified as such below in one of the format versions.

8.1.4 Formats

AEDAT 4.0

The latest release of the AEDAT format is 4.0, introduced in July 2019 with the *Dynamic Vision (DV) software platform*.

It uses Google Flatbuffers[146] to serialize data in a convenient and efficient format, with an easy path to extend existing data structures and introduce new ones. Flatbuffers also allow quick and easy support for other languages, such as Python or Java, to be added, by auto-generating the appropriate support files for them.

All Flatbuffers are size-prefixed, meaning the first four bytes represent a 32 bit little-endian integer encoding the size of the following, actual Flatbuffer data.

All timestamps inside the data are 64 bit integers, representing Unix time[147] in microseconds.

[146] https://google.github.io/flatbuffers/
[147] https://en.wikipedia.org/wiki/Unix_time

Header

- The version header line reads as follows:

```
#!AER-DAT4.0\r\n
```

- A size-prefixed Flatbuffer follows the IOHeader (FB Schema[148]). The IOHeader currently contains the following information:

Field name	Field type	Description
compression	enumeration	Compression algorithm applied to all data streams in the file. Currently supported are: NONE, LZ4, LZ4_HIGH, ZSTD and ZSTD_HIGH. The HIGH compression modes provide smaller file sizes in exchange for higher CPU usage. Scenarios where the highest performance is key should employ LZ4; for a good balance between speed and file size ZST should be used.
dataTablePosition	int64	Offset in bytes of the FileDataTable, containing all the information required to quickly interpret and jump around inside the file. '-1' means no table present.
infoNode	string	Dump of the XML node describing all the data streams and their information (sizes, sources, …). Each node represents a stream, with its name being an integer that is used as StreamID in the rest of the file's data.

- After the header, the actual data streams can be found.

Data

Data packets consist of a PacketHeader (FB Schema[149], fixed 8 bytes struct) which identifies the stream (StreamID integer) and the size of the following content (Size integer).

The content can be compressed or not. If it is compressed (compression != NONE), the whole data block should be fed to the appropriate decompressor for either LZ4 or ZSTD's frame format (each packet is one compressed frame). The result of decompression (or directly the content if compression == NONE) is a size-prefixed Flatbuffer. You can use 'flatbuffers::BufferHasIdentifier()' or similar to get the four character type identifier and parse the content using the appropriate Flatbuffers functions in your language of choice.

FileDataTable

After the data section, there can be one more Flatbuffer, the FileDataTable (FB Schema[150]), that contains information on all the data packets written into the file previously. If the 'dataTablePosition' field in the IOHeader has a value of '-1' this table is not present in this file, else it starts at the given position (offset in bytes). No more data is present after that offset in the file.

AEDAT 3.1

The file starts with several required header lines, as specified *here*. The *version header line* is always the first one, followed by the 3.1 format header line, followed by any others and last the header end line.

After the header, a series of typed event packets, which contain in turn all the various events, are written out.

[148] https://gitlab.com/inivation/dv/dv-runtime/blob/master/modules/output/IOHeader.fbs
[149] https://gitlab.com/inivation/dv/dv-runtime/blob/master/modules/output/FileDataTable.fbs
[150] https://gitlab.com/inivation/dv/dv-runtime/blob/master/modules/output/FileDataTable.fbs

All integer data and fields are always signed and little-endian. This is a departure from previous AEDAT formats, motivated by the fact all the systems we support, currently x86(_64) and ARM, are in fact native little-endian systems, and doing so avoids unnecessary conversion operations.

Header Lines

- Version header line (required): described *here*, always the first header line. Specifically, the line is:

```
#!AER-DAT3.1\r\n
```

- Format header line (required): follows right after the version line, looking like this:

```
#Format: <FORMAT>\r\n.
```

  The *Format* header line describes how the event packets have to be interpreted later on, allowing for optimizations and extensions to be added as needed. Possible formats are: **RAW** (ID=0x00, default).

- Source Identifier header line (required): human-readable identifiers for all event source IDs present in the file are a mandatory part of the header. The corresponding header line shall look like this:

```
#Source <ID>: <DESCRIPTION>\r\n
```

  If multiple sources are present, they must be placed in increasing order by numerical ID (0, 1, 2, 3, …). The description part must be an element from the *list of supported devices*. When re-logging or re-transmitting data, new source headers have to be created representing the new file or network source. The old source headers have to be preserved always, by adding a ***'-' (minus)*** sign in front of them.

```
Example: #-Source 0: DVS128\r\n
```

- Start Time header line (required): this header line encodes the time at which we started transmitting or logging data. The format is the following:

```
#Start-Time: %Y-%m-%d %H:%M:%S (TZ%z)\r\n
```

Time is encoded according to the C strftime() function, see 'man strftime[151]'.

- End of header line (required): follows right after all the other header lines, looking like this:

```
#!END-HEADER\r\n
```

  This allows to clearly determine where the file header ends and data starts.

Event Packets

Each event packet is made up of a common header, followed by its specific event data.

This is not the same header as above, which is placed at the start of a file!

This header is specific to each and every event packet.

Header

The common header has a constant size of 28 bytes and the following format:

[151] http://man7.org/linux/man-pages/man3/strftime.3.html

Bytes	Meaning	Description
0-1	eventType	Numerical type ID, unique to each event type (see 'Event Types' table).
2-3	eventSource	Numerical source ID, identifies who generated the events inside a system.
4-7	eventSize	Size of one event in bytes.
8-11	eventTSOffset	Offset from the start of an event, in bytes, at which the main 32 bit time-stamp can be found.
12-15	eventTSOverflow	Overflow counter for the standard 32bit event time-stamp. Used to generate the 64 bit time-stamp.
16-19	eventCapacity	Maximum number of events this packet can store. **This always equals eventNumber in files and streams, it can only have a different value for in-memory packets.
20-23	eventNumber	Total number of events present in this packet (valid + invalid).
24-27	eventValid	Total number of valid events present in this packet.

By multiplying ***eventCapacity*** with ***eventSize***, and adding the ***28 bytes of header size***, you can quickly and precisely calculate the total size of an event packet.

Ordering

Event packets must be ordered in such a way to guarantee timestamp monotonicity. This means that for all packets of a same type and source, the timestamps must always be monotonically increasing. Further, between packets of different types belonging to the same source, the main timestamp of the first event decides the order in which packets should be written. If this timestamp is equal between different packets, they can be ordered by increasing event type ID, but this is not a requirement, only a suggestion.

Timestamp monotonicity does not need to be guaranteed globally between multiple sources, as any one of them can operate on different time-scales. No order is enforced between sources.

While this ordering requires some additional processing when writing, it greatly simplifies reading, since one always just has to read up to the next packet with a greater first timestamp than whatever timestamp one is interested in, and is guaranteed to have seen all relevant packets with events up to that point.

The **FRAME_EVENT** packet type has four timestamps defined. Of these, "End of Frame" is considered the main timestamp for the purpose of ordering. More generically, the timestamp that is indicated by the **'eventTSOffset'** header field is the one relevant for ordering.

If timestamp reset events occur during a recording (because of device synchronisation or manual reset), then packet timestamp order is no longer monotonic as subsequent timestamps start again from zero. A timestamp reset event of type **SPECIAL_EVENT** will indicate such a reset clearly.

Events

The first 100 event type IDs are reserved for expansion of this format. If you create event types within a private distribution, we recommend that you number them from 100 upwards.

The following event types are supported by default:

Even ID	Event	Description
0	SPE-CIAL_I	Encodes special occurrences, such as timestamp related notifications or external input events.
1	PO-LAR-ITY_EV	Contains change information, with an X/Y address and an ON/OFF polarity. The (0, 0) address is in the upper left corner (standard computer graphics format).
2	FRAMI	Encodes intensity frames, like you would get from a normal APS camera. It supports multiple channels for color, as well as multiple Regions of Interest (ROI). The (0, 0) pixel is in the upper left corner (standard computer graphics format).
3	IMU6_I	Contains data coming from the Inertial Measurement Unit chip, with the 3-axes accelerometer and 3-axes gyroscope. Temperature is also included.
4	IMU9_I	Contains data coming from the Inertial Measurement Unit chip, with the 3-axes accelerometer and 3-axes gyroscope. Temperature is also included. Further, 3-axes from the magnetometer are included, which can be used to get a compass-like heading.
12	SPIKE_	Encodes events from a Dynap-se chip. This event contains information about which chipid generated the event, as well as which core id and which neuron. These addresses are available if the user configure the SRAM of the Dynap-se to route out spikes. Therefore the user can mask or route devices depending on the routing scheme that he decides to use.

Further, the first bit (bit 0) of the first byte of any event is reserved to be a validity bit. This bit is used to mark whether an event is still valid or not, and can be used to efficiently filter out events from a packet. '0' in the 0th bit of the first byte means invalid, '1' means valid. This way zeroing-out an event packet sets all its events to invalid. Care must be taken to put the field containing the validity mark always as the first member of an event.

Also, to generate the full 64 bit timestamp, which doesn't suffer from wrap-around problems, the packet-level 32 bit timestamp (***eventTSOverflow***) and the event-level 32 bit timestamp have to be composed as follows:

```
fullTS = (packet.eventTSOverflow << 31) | event.timestamp
```

(The shift is of 31 bits rather than 32 because of the signed representation).

The various event types and their precise encoding are described in the following sections.

Special Event

Bytes	Meaning	Description
0-3	32 bit data	Holds information on the special event.
4-7	32 bit timestamp	Event-level microsecond timestamp.

Bytes 0-3 are divided in the following way:

Bits	Description
0	Validity mark
1-7	Type of special event (up to 128 types supported)
8-31	Optional data (up to 24 bits)

Currently, the following special event types are defined:

Type ID	Type	Description
0	TIMES-TAMP_WRAP	A 32 bit timestamp wrap occurred. No 'optional data' present.
1	TIMES-TAMP_RESET	A timestamp reset occurred. No 'optional data' present. **Note:** Timestamp reset will have the highest possible timestamp value, to ensure it is always the last event of an event stream, and acts thus as a separator between the old and the new event time stream.
2	EXTER-NAL_IN-PUT_RIS-ING_EDGE	A rising edge was detected (External Input module on device). No 'optional data' present.
3	EXTER-NAL_IN-PUT_FALLING_I	A falling edge was detected (External Input module on device). No 'optional data' present.
4	EXTER-NAL_IN-PUT_PULSE	A pulse was detected (External Input module on device). No 'optional data' present.
5	DVS_ROW_ONL'	A DVS row-only event was detected (a row address without any following column addresses). 'Optional data' is present, encoding the address of the row that generated this DVS row-only event.
6	EXTER-NAL_IN-PUT1_RIS-ING_EDGE	A rising edge was detected (External Input module on device, optional Detector 1). No 'optional data' present.
7	EXTER-NAL_IN-PUT1_FALLING_	A falling edge was detected (External Input module on device, optional Detector 1). No 'optional data' present.
8	EXTER-NAL_IN-PUT1_PULSE	A pulse was detected (External Input module on device, optional Detector 1). No 'optional data' present.
9	EXTER-NAL_IN-PUT2_RIS-ING_EDGE	A rising edge was detected (External Input module on device, optional Detector 2). No 'optional data' present.
10	EXTER-NAL_IN-PUT2_FALLING_	A falling edge was detected (External Input module on device, optional Detector 2). No 'optional data' present.
11	EXTER-NAL_IN-PUT2_PULSE	A pulse was detected (External Input module on device, optional Detector 2). No 'optional data' present.
12	EXTER-NAL_GEN-ERATOR_RIS-ING_EDGE	A rising edge was generated (External Input Generator module on device), and an event injected to show this. No 'optional data' present.
13	EXTER-NAL_GEN-ERA-TOR_FALLING_I	A falling edge was generated (External Input Generator module on device), and an event injected to show this. No 'optional data' present.
14	APS_FRAME_ST	An APS frame capture has started (Frame Event will follow). No 'optional data' present.
15	APS_FRAME_EN	An APS frame capture has completed (Frame Event is alongside). No 'optional data' present.
16	APS_EXPO-SURE_START	An APS frame exposure has started (Frame Event will follow). No 'optional data' present.
17	APS_EXPO-SURE_END	An APS frame exposure has completed (Frame Event will follow). No 'optional data' present.

Polarity Event

Bytes	Meaning	Description
0-3	32 bit data	Holds information on the polarity (luminosity change) event.
4-7	32 bit timestamp	Event-level microsecond timestamp.

Bytes 0-3 are divided in the following way:

Bits	Description
0	Validity mark
1	Polarity (luminosity change): '1' means increase (ON), '0' means decrease (OFF).
2-16	Y event address, up to 15 bits. (0, 0) in upper left corner of screen.
17-31	X event address, up to 15 bits. (0, 0) in upper left corner of screen.

Frame Event

Bytes	Meaning	Description
0-3	32 bit data	Holds information on the frame event.
4-7	32 bit Start of Frame Capture timestamp	Event-level microsecond Start of Frame Capture timestamp.
8-11	32 bit End of Frame Capture timestamp	Event-level microsecond End of Frame Capture timestamp. NOTE: This timestamp is considered the primary timestamp for the purpose of ordering packets.
12-15	32 bit Start of Exposure timestamp	Event-level microsecond Start of Exposure timestamp.
16-19	32 bit End of Exposure timestamp	Event-level microsecond End of Exposure timestamp.
20-23	X length	X axis length in pixels.
24-27	Y length	Y axis length in pixels.
28-31	X position	X axis position (upper left offset) in pixels.
32-35	Y position	Y axis position (upper left offset) in pixels.
36-End of Event	Pixels	Pixel array, 16 bit unsigned integers, normalized to 16 bit depth. Values represent intensity at that pixel and are ready for direct display. First pixel (0, 0) is in upper left corner of screen. Pixels are laid out row by row (increasing X axis), going from top to bottom (increasing Y axis). While the pixel array goes on until the end of the Frame event, only the pixels up to ('X length' * 'Y length' * 'Channel number') are actually valid and contain relevant data. The rest have a value of zero and should not be accessed.

Bytes 0-3 are divided in the following way:

Bits	Description
0	Validity mark
1-3	Color channels number, to track multiple color channels, for example RGB. Valid values are: **GRAYSCALE(1)**, **RGB(3)** and **RGBA(4)**.
4-7	Color filter information, used for interpolating color images (demosaicing). Valid values are: **MONO(0)**, **RGBG(1)**, **GRGB(2)**, **GBGR(3)**, **BGRG(4)**, **RGBW(5)**, **GRWB(6)**, **WBGR(7)**, **BWRG(8)**.
8-14	ROI (Region of Interest) identifier (up to 128).
15-31	Reserved for future expansion.

A frame is considered to be one event - a "frame event". A frame packet may contain 1 or more frame events. Multiplying **eventSize** by **eventCapacity** still gives the length of the data portion of the packet in bytes.

The Frame event is slightly different in that it doesn't have a constant, fixed size across all events, due to the pixel data which may vary due to ROI readouts or different resolutions.

Within a Frame event packet, all Frame events do have the same, constant size, as given by the **eventSize** field in the event packet header. This is as expected and follows the usual scheme.

As such, it's a simple matter to look at the ***eventSize*** field of a Frame event packet's header to figure out the exact size of a Frame event for that particular packet, and thus where each Frame event begins.

Within a Frame event, only pixels up to ('X length' * 'Y length' * 'Channel number') can be accessed with the expectation of having valid values. There might be more pixels than that, determined by how much memory was allocated for the Frame event, but those will all be zero, if they do exist.

Frame Events can directly be used with standard libraries such as OpenCV. For example the following code can be used to generate an OpenCV Mat '**frameMat**' from a Frame Event '**f**', sharing the same pixel array memory:

```
Size frameSize(caerFrameEventGetLengthX(f),
caerFrameEventGetLengthY(f));

Mat frameMat(frameSize,
CV_16UC(caerFrameEventGetChannelNumber(f)),

caerFrameEventGetPixelArrayUnsafe(f));
```

IMU 6-axes Event

The X, Y and Z axes are referred to the camera plane. X increases to the right, Y going up and Z towards where the lens is pointing. Rotation for the gyroscope is counter-clockwise along the increasing axis, for all three axes.

Bytes	Meaning	Description
0-3	32 bit info	Holds information on the special event. **This is needed right now only to hold the validity mark, and is sized to 4 bytes for performance/alignment reasons.**
4-7	32 bit times-tamp	Event-level microsecond timestamp.
8-11	accel_x (float)	Acceleration in the X axis, measured in g (9.81m/s^2).
12-15	accel_y (float)	Acceleration in the Y axis, measured in g (9.81m/s^2)..
16-19	accel_z (float)	Acceleration in the Z axis, measured in g (9.81m/s^2).
20-23	gyro_x (float)	Rotation in the X axis, measured in °/s.
24-27	gyro_y (float)	Rotation in the Y axis, measured in °/s.
28-31	gyro_z (float)	Rotation in the Z axis, measured in °/s.
32-35	temp (float)	Temperature, measured in °C.

Bytes 0-3 are divided in the following way:

Bits	Description
0	Validity mark
1-31	Reserved for future expansion

All floating point values are in IEEE 754-2008 binary32 format, little endian.

IMU 9-axes Event

The X, Y and Z axes are referred to the camera plane. X increases to the right, Y going up and Z towards where the lens is pointing. Rotation for the gyroscope is counter-clockwise along the increasing axis, for all three axes.

Bytes	Meaning	Description
0-3	32 bit info	Holds information on the IMU 6-axes event. **This is needed right now only to hold the validity mark, and is sized to 4 bytes for performance/alignment reasons.**
4-7	32 bit times-tamp	Event-level microsecond timestamp.
8-11	accel_x (float)	Acceleration in the X axis, measured in g (9.81m/s^2).
12-15	accel_y (float)	Acceleration in the Y axis, measured in g (9.81m/s^2)..
16-19	accel_z (float)	Acceleration in the Z axis, measured in g (9.81m/s^2).
20-23	gyro_x (float)	Rotation in the X axis, measured in °/s.
24-27	gyro_y (float)	Rotation in the Y axis, measured in °/s.
28-31	gyro_z (float)	Rotation in the Z axis, measured in °/s.
32-35	temp (float)	Temperature, measured in °C.
36-39	comp_x (float)	Magnetometer X axis, measured in µT (magnetic flux density).
40-43	comp_y (float)	Magnetometer Y axis, measured in µT (magnetic flux density).
44-47	comp_z (float)	Magnetometer Z axis, measured in µT (magnetic flux density).

Bytes 0-3 are divided in the following way:

Bits	Description
0	Validity mark
1-31	Reserved for future expansion

All floating point values are in IEEE 754-2008 binary32 format, little endian.

Spike (Dynap-SE) Event

Bytes	Meaning	Description
0-3	32 bit data	Holds information on the spike event from the Dynap-se device.
4-7	32 bit timestamp	Event-level microsecond timestamp.

Bytes 0-3 are divided in the following way:

Bits	Description
0	Validity mark
1-5	Core id, this is the virtual_core_id programmed in the SRAM cell of the Dynap-se (bits <28:29>)
6-11	Chip id, this is the destination_core_id programmed in the SRAM cell of the Dynap-se (bits <18:21>)
12-31	Neuron id, the source neuron address

Formats

Here the specifics about the event packet encoding formats, as given by the Format header line:

RAW (ID=0x00)

A direct dump of the memory holding an event packet. No processing is needed at input, allocating the right amount of memory and reading the bytes into it directly is sufficient.

AEDAT 3.0

This section explains how version 3.0 differed from version 3.1.

Header Lines

- The version header line reads as follows:

```
#!AER-DAT3.0\r\n
```

- There is no end of header line required. Therefore, the only way to check where the header ends is to check that a line starts with the '**#**' character and then read the line up to the new-line marker, but this can fail if the very first byte of the first event also has a value of 35 (decimal), corresponding to the '**#**' character.

Polarity Event

The origin point is in the lower left of the screen rather than the upper left, meaning that it is not compatible with the standard computer graphics way of doing things, e.g. libraries such as OpenCV.

Frame Event

The origin point is in the lower left of the screen rather than the upper left, meaning that it is not compatible with the standard computer graphics way of doing things, e.g. libraries such as OpenCV.

The color filter enumeration values are different due in part to a smaller available range; the ROI identifier is thus positioned differently. Both features were never in active use for AEDAT 3.0:

Bytes	Meaning	Description
0-3	32 bit info	Holds information on the frame event.

Bytes 0-1 are divided in the following way:

Bits	Description
0	…
1-3	…
4-6	Color filter information used for interpolating color images (demosaicing). Valid values are: **MONO(0)**, **RGBG(1)**, **RGBW(2)**.
7-13	ROI (Region of Interest) identifier (up to 128).

New Event Types

The following event types were not implemented in version 3.0:

Spike

AEDAT 2.0

Version 2.0 was introduced in 2010. It is equivalent to version 1.0, but widens the address field to 32 bit. It also requires the *header* version line to disambiguate between formats. Then again, an optional header, followed by a series of [address, timestamp] pairs, each making up an event. The address is 32 bit wide, and the timestamp is also 32 bit, a total of 8 bytes per event. The timestamp is in microseconds, while the address has to be interpreted according to a specific jAER AEChip class' definition of that address. This could be the DVS128 class, the same as *AEDAT 1.0*, or the DAVIS family of classes (DAVIS240A,B,C; DAVIS346cBSI, DAVIS640, …). All integer data and fields are always signed and big-endian!

The jAER software, since 2014, also writes several informative header lines, these include the ***'HardwareInterface'*** and the ***'AEChip'*** header line contains information on the current device and the class interpreting its output. Here is an example of these informational lines:

```
# This is a raw AE data file - do not edit

# Data format is int32 address, int32 timestamp (8 bytes total),
peated for each event

# Timestamps tick is 1 us

# created Thu Dec 03 14:47:00 CET 2015

# HardwareInterface: DAVIS FX3 0002-INI

# AEChip: eu.seebetter.ini.chips.davis.Davis640
```

Further, a full XML-like dump of all the preferences for the above AEChip class, which includes settings such as the APS exposure time or the biases, is generated, and kept either fully in the header as header lines or, in the future, in a separate XML file, which is named

```
<filename>-prefs.xml
```

and is referenced in the header as one header line as follows:

```
# Prefs-File: <filename>-prefs.xml\r\n
```

Entries look like the following in XML (the example is of an addressable bias):

```
<entry key="DAVIS240C.AddressedIPotCF.ApsROSFBn.LowCurrent" value="false"/>
```

Ordering

All events in the AEDAT 2.0 format are ordered by their timestamp, and should guarantee timestamp monotonicity, meaning the next event will always have an equal or greater timestamp.

It is possible to find older files where this was not guaranteed due to older hardware and its logic sometimes not ensuring that timestamp increase events be always delivered. When this is the case, the software will detect this, warn the user and continue working by "jumping back" to the new but older timestamp value.

DAVIS

The DAVIS camera family stores polarity (luminosity change) events, IMU (Inertial Measurement Unit) samples and pixel intensity values (both APS reset and signal read) according to the following scheme:

Bit 31

Bits	Meaning	Description
31	Type	Defines the type of address stored here. '0' means DVS, '1' means APS or IMU (see bits 11-10).

Bit 30-12

IMU:

Bits	Meaning	Description
30-28	IMU sample type	Type of IMU sample: 0 -> Accel X 1 -> Accel Y 2 -> Accel Z 3 -> Temperature 4 -> Gyro X 5 -> Gyro Y 6 -> Gyro Z
27-12	IMU sample	For IMU events, 7 words are sent in series, these being: - 3 axes for accel, - Temperature, - 3 axes for gyro.

DVS or APS:

Bits	Meaning	Description
30-22	Y address	Y event address. (0, 0) in the lower left corner of the screen.
21-12	X address	X event address. (0, 0) in the lower left corner of the screen.

Bit 11-10

APS:

Bits	Meaning	Description
11-10	sub-Type	00 -> APS Reset Read 01 -> APS Signal Read 10 -> Unused 11 -> IMU Sample

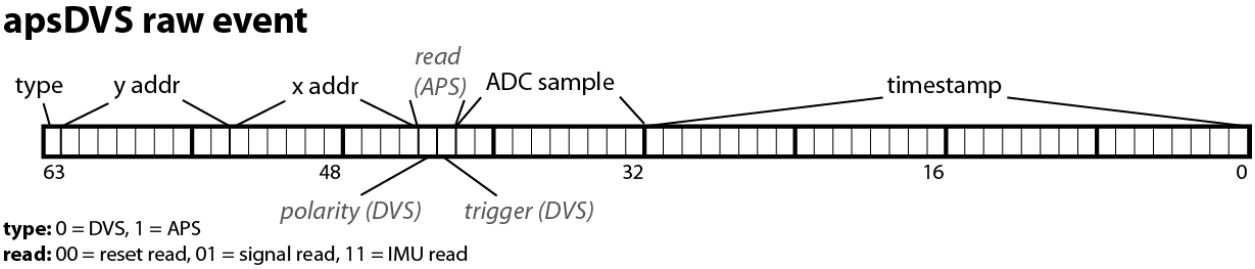
DVS:

Bits	Meaning	Description
11-10	sub-Type	00 -> DVS Polarity OFF 01 -> External Event (same as 11) 10 -> DVS Polarity ON 11 -> External Event (Same as 01)

Bit 9-0

Bits	Meaning	Description
9-0	ADC sample	10-bit ADC sample representing pixel intensity. Only for Type=APS, else zero.

The following image explains the same thing in a more graphic way (IMU data is not shown in this graphic, however):



Frames, in this format, are laid out as a sequence of events, one for each pixel. The timestamp for each pixel is also stored and corresponds to the start of the readout of the column where the pixel resides. To get start and end of frame and exposure timestamps, one has to look at the timestamp of the pixel readouts closest to those moments:

- Start of Frame: first reset read pixel.
- Start of Exposure: last reset read pixel for GlobalShutter mode, first reset read pixel for RollingShutter mode.
- End of Exposure: first signal read pixel.
- End of Frame: last signal read pixel.

DAS1

The DAS1 (Dynamic Audio Sensor, CochleaAMS1c AEChip class) stores its data according to the following format:

Bit 13-10

Bits	Meaning	Description
13	Type (ADC or AER)	Defines the type of address stored here. '0' means AER from Cochlea chip, '1' means sample from external ADC.
12	ADC scanner sync	Signals if the current ADC sample is newly synchronized with the Scanner. If '1', it is, so we can clear and reset buffers. If '0', just get the ADC data.
11-10	ADC channel	Defines which of the four possible ADC channels this sample belongs to.
9-0	ADC sample or AER address	If Type=ADC, then this contains the 10-bit ADC sample ; If Type=AER, then this is the 10-bit AER address from the Cochlea chip. The 10 bits are to be interpreted this way:

Bits from 9 to 0 are divided in the following way:

Bits	Description
9-8	Neuron number (0 to 3).
7-2	Channel number (from 0 to 63, where 0 is the highest frequency, 63 is the lowest frequency).
1	Left/Right ear ('0' is left, '1' is right).
0	Neuron bank ('0' if from BPF, '1' if from SOS).

Note that although the DAS1 has synchronization ports, the insertion of external trigger events into the event stream via a synchronization pulse has never been implemented in the firmware.

AEDAT 1.0

Version 1.0 of the AEDAT format was the first version implemented in jAER in 2008. It simply consists of an optional *header*, followed by a series of [address, timestamp] pairs, each making up an event. The address is 16 bit wide and the timestamp 32 bit, a total of 6 bytes per event. The timestamp is in microseconds, while the address has to be interpreted according to a specific jAER AEChip class' definition of that address. All integer data and fields are always signed and big-endian!

This format is deprecated and should not be used for any new development or recording! At most, support reading this format.

Ordering

All events in the AEDAT 1.0 format are ordered by their timestamp and should guarantee timestamp monotonicity, meaning the next event will always have an equal or greater timestamp.

It is possible to find older files where this was not guaranteed due to older hardware and its logic sometimes not ensuring that timestamp increase events be always delivered. When this is the case, the jAER software will detect this, warn the user and continue working by "jumping back" to the new but older timestamp value.

DVS128

For version 1.0, this was usually the DVS128 chip class, which uses the following format to store polarity (luminosity change) events generated by the device:

Bits	Meaning	Description
15	External event	External event detected on the IN pin (TS-Master mode).
14-8	Y address	Y event address. (0, 0) in the lower left corner of the screen.
7-1	X address	X event address. (0, 0) in the lower left corner of the screen.
0	Polarity	Polarity (luminosity change): '1' means increase (ON), '0' means decrease (OFF).

8.1.5 Network Streaming

Streaming data over the network, instead of to a file, is done with exactly the same data formats.

The only difference is that with streaming, no *headers as defined above* are sent, meaning receivers need software switches to understand what they are receiving and how to interpret it. In jAER, for versions 1.0 and 2.0, those can be found in the network streaming dialogues, as well as by selecting the appropriate AEChip class manually for data parsing.

Version 3.X, on the other hand, attempts to automate this by sending a 20 bytes network header that signals the presence of version 3.X data with a magic number at the start.

For stream based protocols, like TCP, this header will be sent once at the start of the stream.

For message-based network communication, for example UDP, this header will be part of each message, at the front, to ensure it can be properly parsed always.

All header fields/integers are little-endian.

The 20 bytes network header format is the following:

Byte	Meaning	Description
0-7	64bit AEDAT 3.X magic number	Magic number signalling presence of AEDAT 3.X header and data: **0x1D378BC90B9A6658.**
8-15	Sequence Number	Increasing 64 bit integer to detect missing message parts. **Not used for stream protocols (TCP), set to zero.**
16	Version Number	The AEDAT 3.X version number part. 0x00 for AEDAT 3.0, 0x01 for AEDAT 3.1.
17	Format Number	The numerical ID representing the used *format*.
18-19	Source ID	Source ID for all the events of this network stream. One network stream always carries data from only one Source ID on the sender system.

8.1.6 Supported Devices

The following is a list of all supported input devices and their names (as would be written in *AEDAT 3.1* Source ID *header lines*):

Device Name	Historical Alternative Names
File	
Network	
DVS128	Tmpdiff128
DAVIS240A	SBret10
DAVIS240B	SBret20
DAVIS240C	SBret21, DVS240
DAVIS128	Davis128Mono, Davis128Rgb
DAVIS208	Davis208Mono, Davis208Rgbw, PixelParade, SenseDavis192, SensDavis192
DAVIS346A	Davis346AMono, Davis346ARgb
DAVIS346B	Davis346BMono, Davis346BRgb, Davis346
DAVIS346Cbsi	Davis346bsi
DAVIS640	Davis640Mono, Davis640Rgb
DAVISHet640	DavisHet640Mono, DavisHet640Rgbw, CDavis, CDavis640, Davis640Rgbw, CDavis640Rgbw
DYNAPSE	Dynap-se, DYNAPSEFX2
DVXplorer	DVXplorer Lite

These Historical Alternative Names are names which devices may have been called in the past (not including different capitalisations), these are not to be used for Source ID headers, they are only here as a reference!

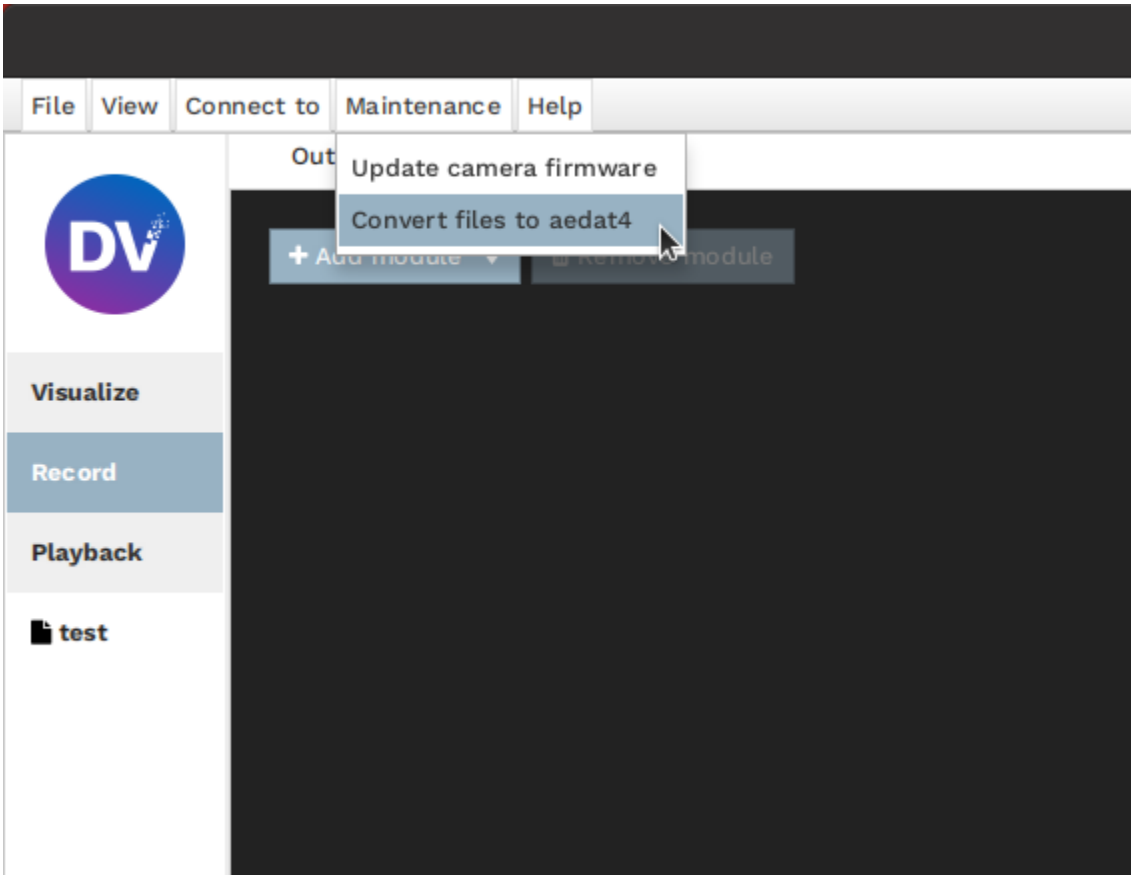
8.2 Convert AEDAT2/3 Files to AEDAT4

AEDAT is the standard file format used for storing data generated from our event cameras. AEDAT files can be used to record and playback different information captured by our event cameras, including pixel value changes, frames, and IMU data.

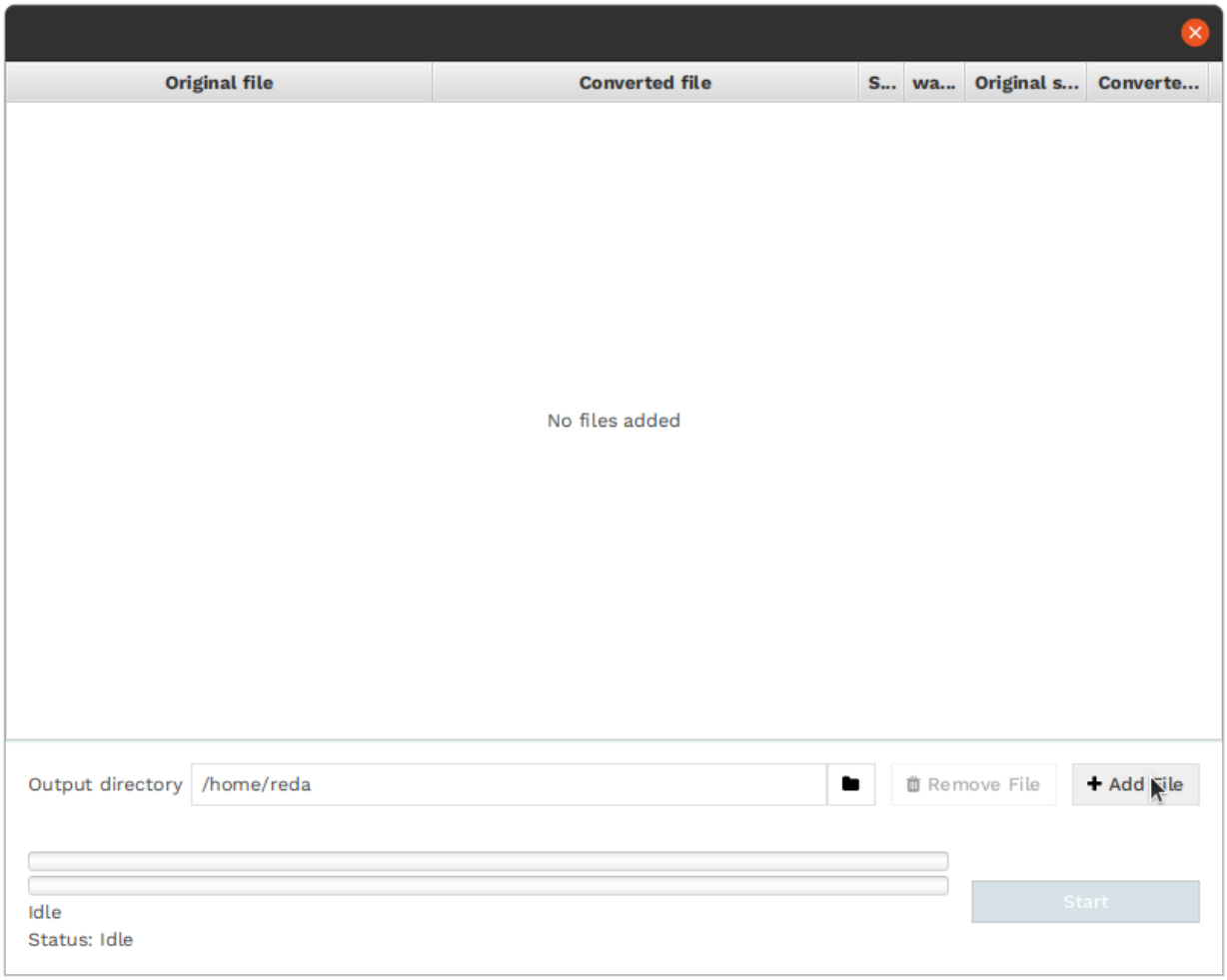
Over the years, we progressively updated the AEDAT file format for our output files. The latest version is number 4, which is currently used by the DV-SDK. To continue using older AEDAT2 and AEDAT3 files, please convert them to the AEDAT4 format.

Converting old formats using ***DV*** is straightforward. Just follow the steps below:

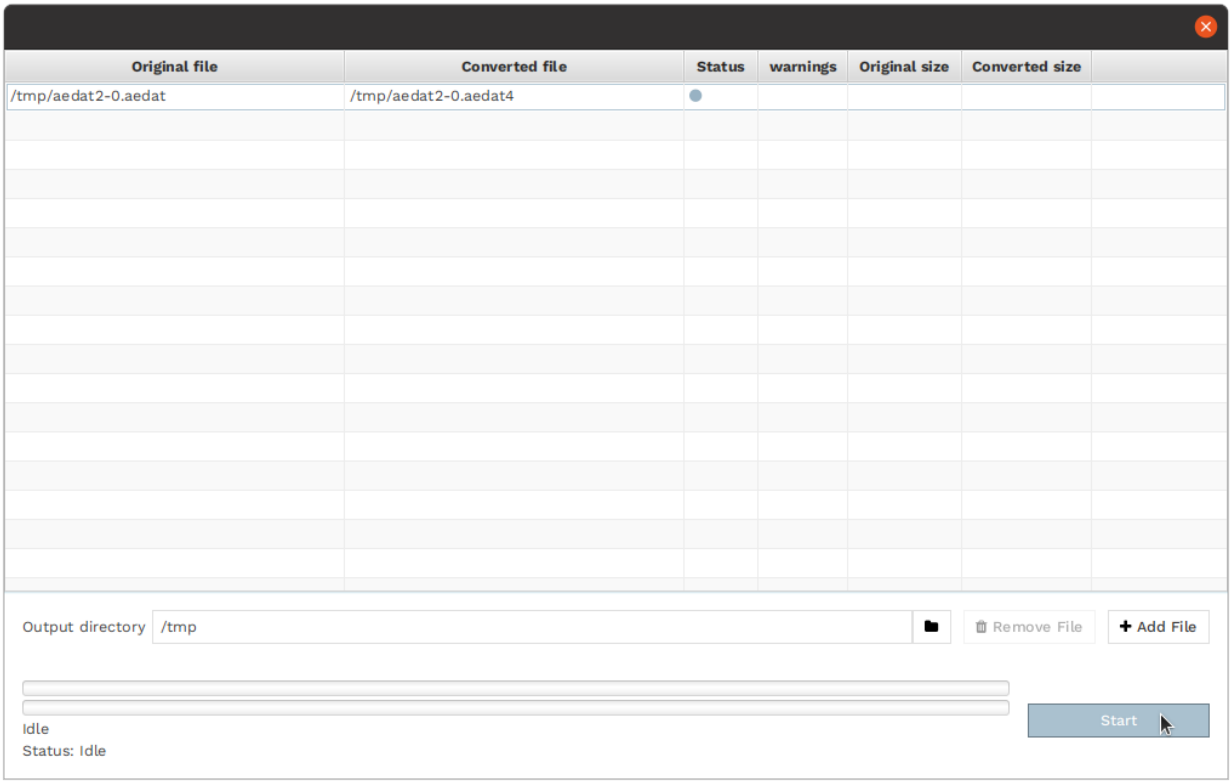
- Go to `Maintenance` (top menubar in DV) and select `Convert files to aedat4`



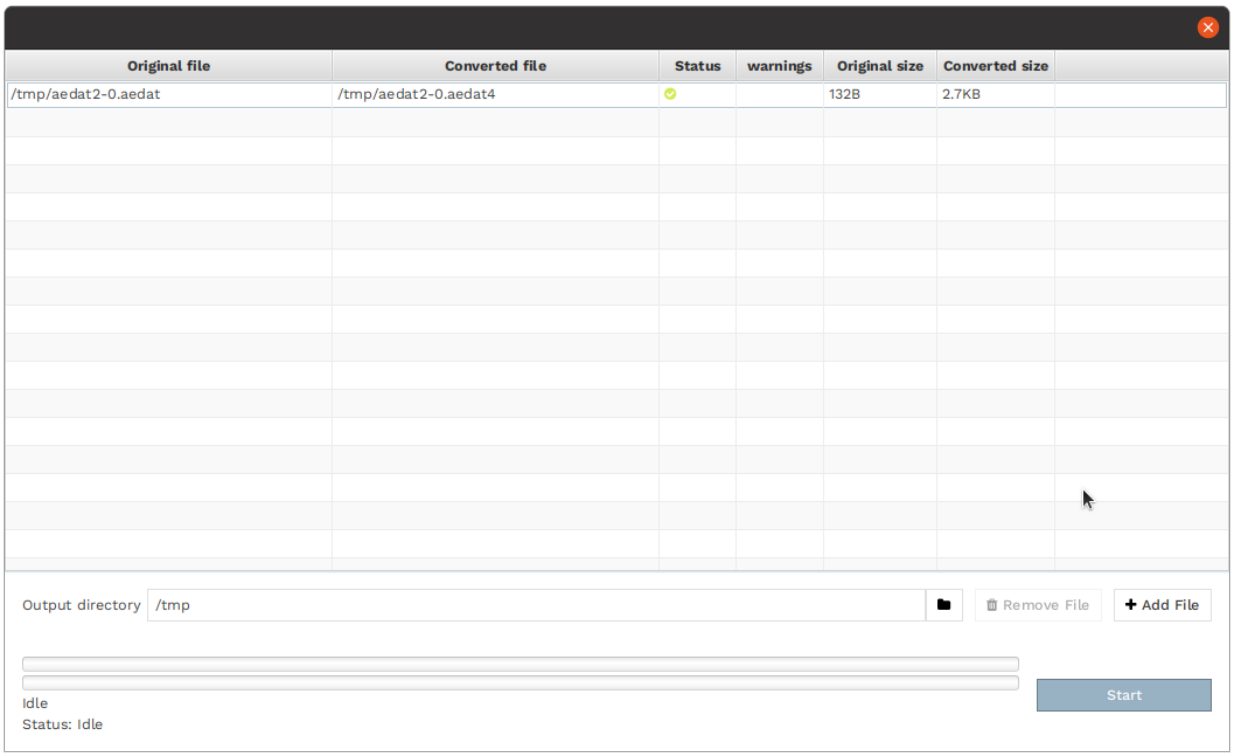
- Add the files you want to convert



- Select the output directory and then press `Start`



- After you press `Start` all the files added will be converted to AEDAT4. When the conversion is finished, you should see all files with a green check mark in the status column. If the conversion fails for any reason, a yellow symbol will be displayed, and you will be able to get more details on what went wrong.



Now, you will find all the converted files in the output directory. During the conversion, the converter will keep the filename, but will change the file extension to AEDAT4.

You can now use the standard `File Input` module and functionality with your newly converted files.

8.3 Camera Calibration

> **Note**
>
> To perform camera calibration with DV please follow *these instructions* instead.

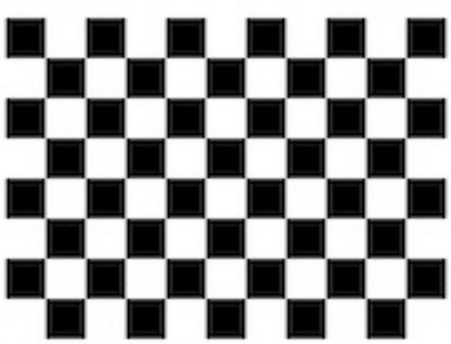
The following instructions target a purely script-based calibration.

8.3.1 Goals

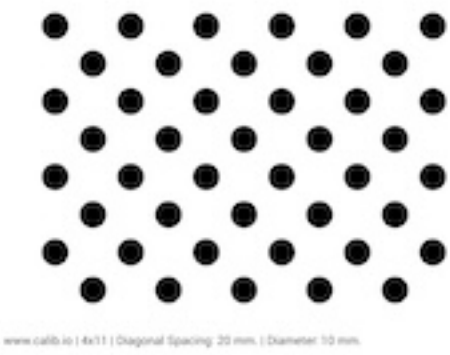
Camera calibration aims to compute calibration parameters. These parameters represent the focal length, image sensor format, and camera principal point. Read more on Wikipedia[152].

8.3.2 Setup

To perform calibration, a calibration pattern is needed. Multiple types of patterns are available, the ones currently supported by this tutorial are:

- April Grid 
- Checkerboard 

[152] https://en.wikipedia.org/wiki/Camera_resectioning#Intrinsic_parameters



- Asymmetric Circles

One can create, download and print the desired pattern using calib.io[153].

8.3.3 Procedure

The only difference to calibrate an event camera compared to a frame camera is the need to *reconstruct frames from events*. The other steps are identical.

Image Reconstruction

We provide a repository[154] with python samples that allow the user to reconstruct images from events using the different state-of-the-art image reconstruction methods. All instructions to perform the desired reconstruction are also provided.

Calibration

The calibration steps are simple:

- Place the calibration pattern in the view of the camera
- Move the calibration pattern in the view of the camera. With the movements, try to cover all the different parts of the camera view and provide varying angles of the pattern with respect to the camera.
- Provide the live or recording image output to the selected calibration pipeline and follow the corresponding instructions.

Using Kalibr

We provide a repository[155] based on the Kalibr library[156] with C++ samples that allow users to perform calibration from frame files, stored in aedat4 format. All instructions to perform the desired calibration are also provided.

Using OpenCV

OpenCV provides multiple tutorials to perform camera calibration from live frames or recorded files. Feel free to use them if you are familiar with OpenCV:

- Python Tutorial[157]
- C++ Tutorial[158]

[153] https://calib.io/pages/camera-calibration-pattern-generator
[154] https://gitlab.com/inivation/dv/modelfactory
[155] https://gitlab.com/inivation/dv/modules/dv-imu-cam-calibration
[156] https://github.com/ethz-asl/kalibr
[157] https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html
[158] https://docs.opencv.org/4.x/d4/d94/tutorial_camera_calibration.html

8.4 Use a Remote DV-Runtime Instance

8.4.1 Start a Remote DV-Runtime Instance

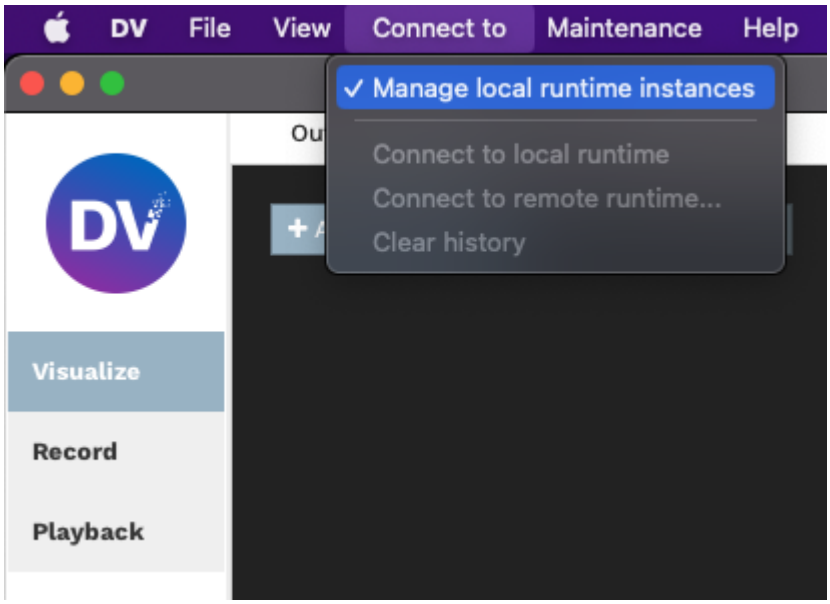
For remote, head-less runtime deployments, we support and recommend using the system service daemons present in all major Linux distributions: SystemD or OpenRC. Our distribution packages will install the necessary control scripts by default, users only have to enable them as needed following the below procedure:

1. Edit the startup scripts at */usr/lib/systemd/system/dv-runtime.service* (SystemD) or */etc/init.d/dv-runtime* (OpenRC). Change the IP address from `127.0.0.1` (listen on local host only) to `0.0.0.0` (listen on all network interfaces).
2. Start the service using the appropriate command:
   - SystemD: `$ sudo systemctl start dv-runtime.service`
   - OpenRC: `$ sudo /etc/init.d/dv-runtime start`
3. Optional: set the service to automatically start on boot:
   - SystemD: `$ sudo systemctl enable dv-runtime.service`
   - OpenRC: `$ sudo rc-update add dv-runtime default`

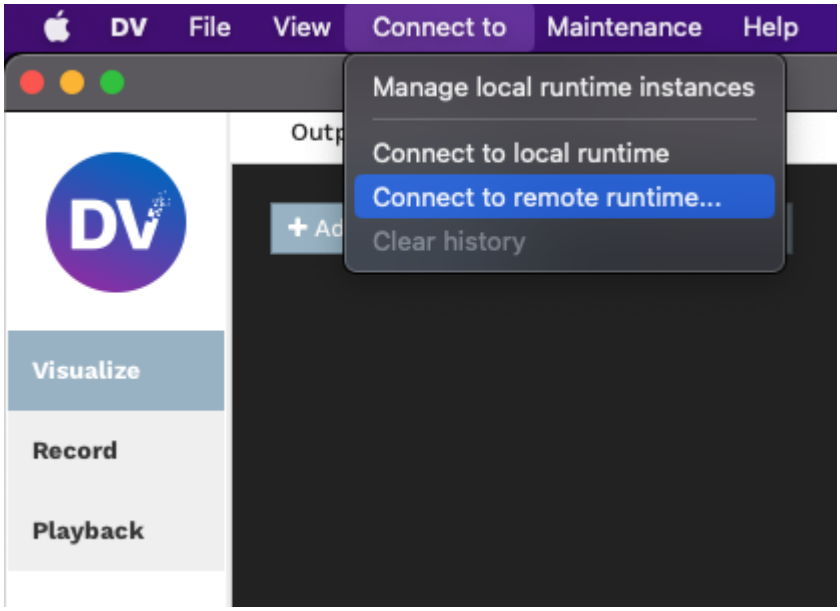
8.4.2 Connect to a Remote DV-Runtime Instance

In DV:

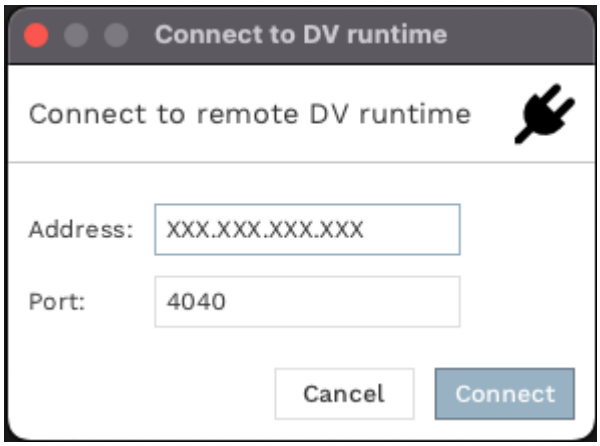
1. Go to 'Connect to' in the top menu and disable the management of local instances.



2. Select 'Connect to remote runtime …'



3. Enter the IP address of the system running the runtime and the port on which the runtime is listening (default is 4040).



4. Press 'Connect' and you're done!

Note1: If the connection does not work, or visualizers fail to show data, check that there are no firewalls or anything else blocking TCP connections. TCP port 4040 is needed for configuration exchange, plus one random TCP port in the ephemeral range[159] for each visualizer to transmit data to show.

Note2: Configuration can also be changed via dv-control[160].

[159] https://en.wikipedia.org/wiki/Ephemeral_port
[160] https://dv-runtime.inivation.com/master/runtime/usage/dv-control.html

CHAPTER

NINE

FREQUENTLY ASKED QUESTIONS

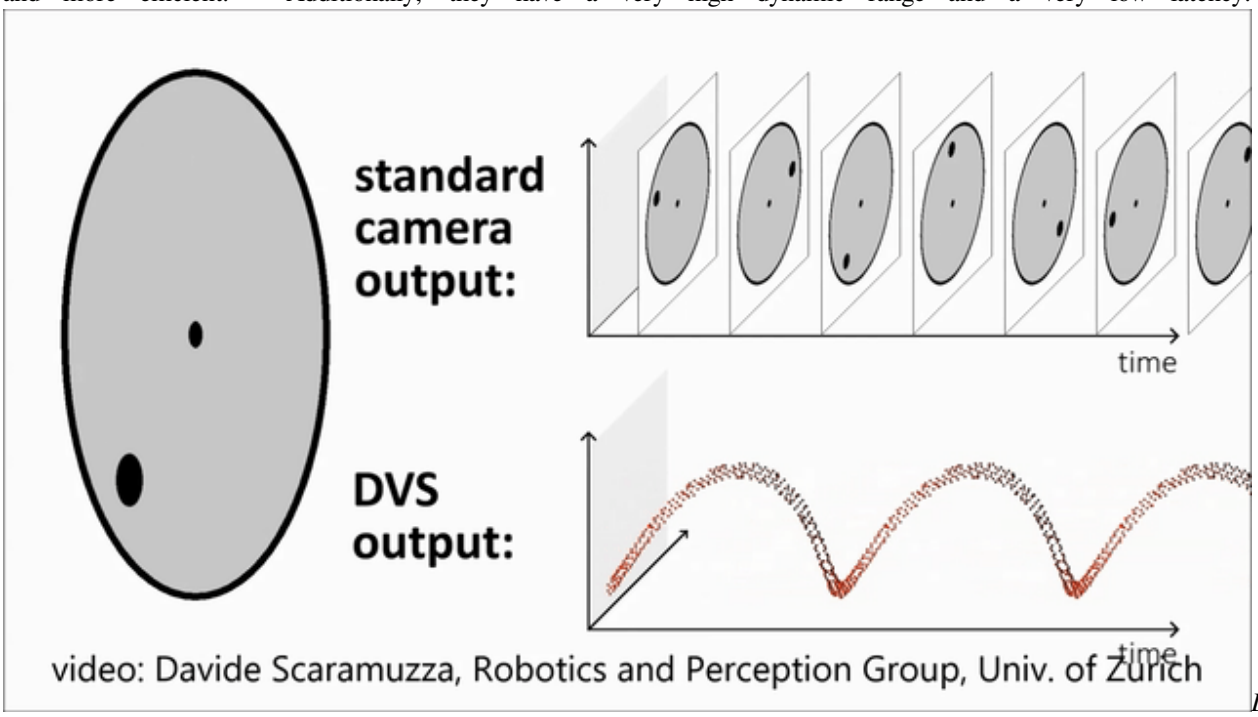
List of Questions

- *Frequently Asked Questions*
  - *General Questions*
    - *What is a DVS?*
    - *How do I use my Event Camera?*
    - *I think I need both events and frames in my applications. Which camera should I choose?*
    - *Can DVS cameras see infrared (IR)?*
  - *Hardware Questions*
    - *My camera is not working, what should I do?*
    - *Does iniVation offer drivers for the cameras?*
    - *Can I use an external trigger signal?*
    - *Can I synchronise timestamps between two cameras?*
    - *What is the spectral sensitivity / quantum efficiency (QE) of the iniVation sensors?*
    - *How can I change the biases of my camera to, for example, increase sensitivity or slow down event output?*
    - *Why does my DAVIS camera give out frames that are all white / have vertical lines / look corrupted?*
  - *Software Questions*
    - *How can I read the raw data from an AEDAT4 file?*
    - *I recorded files in an old Aedat format. Can I use them in DV?*
    - *Can I use Python for prototyping?*
    - *Can I use ROS with iniVation cameras?*
    - *Can I use iniVation cameras and DV with a Raspberry Pi?*
    - *Is it possible to talk to the chip directly using an external FPGA / µc?*
    - *I'm having USB connection issues*
    - *When are timestamps added, and can there be delays?*
    - *Can events be lost?*

9.1 General Questions

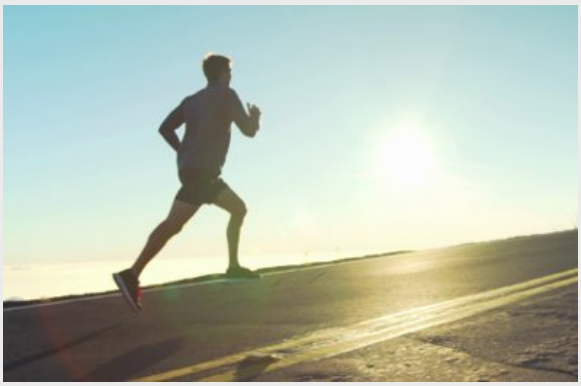
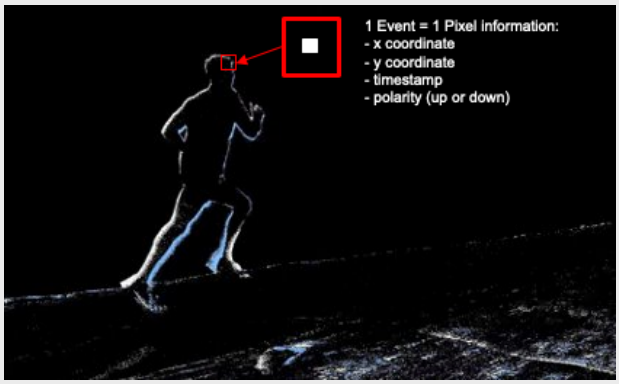
9.1.1 What is a DVS?

DVS (Dynamic Vision Sensor) is a type of sensor that transmits only pixel-level changes. Unlike traditional image sensors that capture full frames at a fixed rate, DVS sensors are much faster and more efficient. Additionally, they have a very high dynamic range and a very low latency.



Davide Scaramuzza, Robotics and Perception Group, University of Zurich

Below is a comparison between a more realistic frame and the corresponding event output with an explanation of what an event is.

Realistic Image from Frame Camera	Corresponding Output from Event Camera
	

Read more about events and temporal resolution in our *White Paper*.

9.1.2 How do I use my Event Camera?

iniVation cameras are usable with our different *software solutions*.

9.1.3 I think I need both events and frames in my applications. Which camera should I choose?

All iniVation cameras provide pixel-level event output. In addition, all iniVation cameras support software reconstructed frames of intensity images, using the event output processed by iniVation's *DV Software*. DV Software contains several algorithms to reconstruct frames of intensity images from events. Each of these algorithms has different characteristics affecting image quality and processor requirements.

In addition, the *DAVIS346 camera* can directly output conventional frames of hardware-measured intensity images. These are the same as the frames from conventional image sensors. The conventional frame output from the DAVIS346 is limited to 55 dB dynamic range and 40 FPS frame rate. Please refer to the *DAVIS346 specifications* for further details.

Software reconstructed frames (from events) work best for specific use cases where image quality is not critical and/or ultra-high speed is required. Some examples of such use cases are as follows:

- General debugging output for users
- Occasional camera calibration, lens focusing, general setup
- Basic recognition tasks where the user is collecting their own training data
- Extremely high-speed visualization that is not easily possible using conventional frames

Comparing to conventional frames, software reconstructed frames have the following pros and cons :

Pros:

- Inherently HDR (>90 dB)
- Variable frame rates (up to approx. 1k FPS depending on available processing power)
- Robust reconstruction depending on the use case
- Can be improved via software updates

Cons:

- Lower image quality than conventional frames
- Image artefacts (reconstruction algorithm dependent)
- Unable to reconstruct static scene information
- Requires extra computational power

If image quality is of primary importance to a certain use case, e.g. difficult image recognition, it is best to check beforehand to see if software reconstructed frames will be of sufficient quality. If you are unsure, please get in touch with our support team[161], and we will provide you with the best camera recommendation for your use case.

9.1.4 Can DVS cameras see infrared (IR)?

DVS sensors can see all visible light plus near-infrared (NIR). The luminosity function should be that of a standard CMOS sensor, sensitivity peaking around 700-750 nm and going until about 1100 nm wavelength (however, the highest wavelengths we tested were around 980nm). The default lens shipped with the camera may contain an IR filter. In order to see IR light, you may need to use a different lens.

[161] support@inivation.com

9.2 Hardware Questions

9.2.1 My camera is not working, what should I do?

If you can't seem to access your camera, please try the actions below:

- Ensure the camera is properly connected with the provided cable or a decent equivalent. Try a different cable.
- Try switching USB ports, preferably other USB 3.0 ports (usually blue).
- Check your USB permissions. Reboot your computer once. On Windows, you may have to *install the drivers manually* in some rare occasions.
- If you are using *DV GUI*:
  - Make sure *DV is up-to-date* to its latest version.
  - Make sure to properly *select the camera* in the list of devices before running any other modules.

If none of this works, then please contact us through our support e-mail[162] so that we can investigate your issue further.

9.2.2 Does iniVation offer drivers for the cameras?

On Linux and macOS, no driver is necessary. The cameras work out of the box.

On Windows, the driver should get installed automatically once you plug in the device. If that doesn't work, install the driver manually by following *this guide*.

9.2.3 Can I use an external trigger signal?

Yes, *DAVIS346* and *DVXplorer* support connecting an external trigger signal. Usage is documented *here*. Please note the *DAVIS346 AER* and the *DVXplorer Micro* do not support external trigger signals.

If you require a sync cable, we recommend the following supplier[163].

9.2.4 Can I synchronise timestamps between two cameras?

Yes, that is one of the purposes of the sync connector available on the *DAVIS346* and *DVXplorer*. Usage is documented *here*. Please note the *DAVIS346 AER* and the *DVXplorer Micro* have no support for multi-camera timestamp synchronization.

9.2.5 What is the spectral sensitivity / quantum efficiency (QE) of the iniVation sensors?

The spectral sensitivity / quantum efficiency (QE) of DAVIS346 mono is published in https://ieeexplore.ieee.org/document/8334288 (FSI DAVIS curve in Fig. 3). The spectral sensitivity / QE of DAVIS346 color is not available, but the QE of the same color filters and similar pixel designs (same photodiode size) have been characterized in this thesis https://www.research-collection.ethz.ch/handle/20.500.11850/156363 (Chapter 5 Fig. 31 B, Page 71). So the QE of DAVIS346 color should be very similar to the R, G, and B (without W) channels in that figure. The QE specs of DVXplorer or DVXplorer Lite are not available. Their QEs are not directly measurable because their pixels have no intensity output.

More information specific to InfraRed light is available *here*.

[162] support@inivation.com
[163] https://www.alibaba.com/product-detail/HR10A-male-and-female-4P-cable_60631952696.html

9.2.6 How can I change the biases of my camera to, for example, increase sensitivity or slow down event output?

Biasing documentation is available *here*.

9.2.7 Why does my DAVIS camera give out frames that are all white / have vertical lines / look corrupted?

DAVIS camera frames are known to have issues:

- All white frames are generally caused by overexposure of the camera. Try to:
  - Not use it while looking directly at a very bright light source.
  - Change the frame exposure settings.
- Vertical lines in the frame might be caused by using the rolling shutter mode. Make sure to use global shutter setting.
- Refer to *this question* to check that you are using your camera properly.

If you still have issues, contact us through our support e-mail[164].

9.3 Software Questions

9.3.1 How can I read the raw data from an AEDAT4 file?

An `.aedat4` file contains data under the *AEDAT4 format*. The data is compressed, it is therefore not directly human-readable. However, there are multiple solutions to extract the raw data from an AEDAT4 file:

- Using dv-processing[165] in C++ or Python.
- Convert the event data to `.csv` using the *corresponding DV module*

9.3.2 I recorded files in an old Aedat format. Can I use them in DV?

DV can convert old AEDAT files. Please check the *corresponding instructions*.

9.3.3 Can I use Python for prototyping?

Yes, you can use *dv-processing* in Python for that purpose.

9.3.4 Can I use ROS with iniVation cameras?

Yes, you can use our *dv-ros package*.

9.3.5 Can I use iniVation cameras and DV with a Raspberry Pi?

Yes, you can use our cameras through DV with a Raspberry Pi.

In order to do that, you need to:

- *Install dv-runtime on your Raspberry Pi*
- *Connect to the runtime of the Raspberry Pi with another computer via DV*

[164] support@inivation.com
[165] https://dv-processing.inivation.com/master/reading_data.html#from-a-file

9.3.6 Is it possible to talk to the chip directly using an external FPGA / µc?

Model	AER pins exposed
DVS128	No
DAVIS240	Yes
DAVIS346	No
DAVIS346 AER	Yes
DVXplorer (Lite)	No
DVXplorer Mini	No
DVXplorer Micro	No

As shown in the table above, this depends on the hardware. Only certain models offer physical access to the AER pins.

On those that do, we only support external control for the *AER DVS events bus*.

To enable this, our own logic has to be instructed to not negotiate the events and to put the ACK pin in high-impedance, to do this disable the 'DVS.Run' configuration option and enable the option 'DVS.ExternalAERControl'. We do not support powering, configuring or biasing the sensor from external systems, nor do we support extracting the APS frame or IMU data. Our USB devices always require a USB connection at least to power and configure the sensor!

9.3.7 I'm having USB connection issues

USB3 can be more tricky than USB2 to get working right, here a few pointers:

1. try another USB3 cable
2. don't use USB3 cables longer than 1m, as longer cables are not standardized and quality varies
3. try the back-ports of your desktop, the front-ports are often badly shielded
4. make sure your BIOS and drivers are up-to-date, there's often updates to the USB controllers included
5. early AMD Ryzen systems are known to have issues with USB devices, a BIOS and driver update will help there (further information[166]). A new BIOS with the proper fixes was released at the start of April 2021 and should be available from most manufacturers.

The cable is really in our experience the most problematic part, but the quality of USB3 ports on computers also varies wildly. For longer cables, we've had good success using this 5m one[167], even in industrial settings.

Very old USB3 chip-sets from when USB3 started out (2011-2013), from Renesas and similar early manufacturers, often were problematic. Nowadays with USB3 directly integrated on the main motherboard chip-sets, and not coming off separate external chips, there are usually far less issues, and in general recent Intel hardware has performed the best. USB3 ports coming off Intel 3/4/500 series chip-sets and newer have generally been very reliable in our experience.

Be careful that sometimes motherboards having lots of USB3 ports don't attach all of them to the main chip-set, but go through alternative paths and chips. The first row of blue USB3 connectors from the top of the back of your PC, usually right after the row of black USB2 still present for keyboard/mouse, is the "best" connected one. Laptops often struggle with more than two high-traffic USB devices, such as cameras, attached.

If nothing above helps, try out a USB2 cable as a last resort. This is always an easy test, and often you might not even need the extra bandwidth USB3 offers if only using events from the camera. If it works fine with USB2, you know at least that the device is working properly and not at fault, and the issue is most probably in one of the above points related to the USB3 support of your computer.

[166] https://www.tomshardware.com/news/amd-suggest-possible-fixes-for-usb-connectivity-issues
[167] https://store.unibrain.com/shop/usb3x-products/usb-3-0-cables/5m-16-4-ft-usb-3-0-cable-type-a-to-micro-b-screw-lock/

9.3.8 When are timestamps added, and can there be delays?

On DVS128 and DAVIS sensors, the sensor chip itself has no concept of time, only of events happening and it trying to get those out as soon as possible, so the timestamps are added after in the FPGA logic.

The timestamping happens in order of how the events are read out from the sensor interface. As soon as an event happens, the sensor tries to send it out, the FPGA reads it and timestamps it. This works well if events are relatively sparse or well distributed; if you instead have pretty much everything turning on and off at the same time, everything tries to queue up to be read out, and due to the X/Y arbitration circuits not being perfectly fair due to process variations, you get a kind of 'scanning effect' where it reads out following a mostly reproducible order. Indeed in this overload case, timestamps "loose" precision, and events might be lost (in the sense that if a pixel is waiting to be read, it cannot generate another event while it's waiting, even if there was a change in the light hitting it). You're hitting the readout bandwidth limit. You can try to slow down the pixels by changing the 'Refr' bias (refractory period of pixels).

In summary: overload -> loss of timing information. In the ideal case, loss would be uniformly distributed thanks to random readout. In reality, loss is unevenly distributed due to (mostly) predictable readout.

It is also possible to loose timing precision if the device side buffers on the FPGA are full because of USB not transferring data (see the *next question*), since the association between timestamp and event happens in a stage after the DVS chip readout one. So events can be read out and queued up in an internal buffer, waiting to be timestamped and then sent out via USB, if USB itself is busy.

On DVXplorer cameras, it depends on the model. The Samsung S5K231Y sensor does internal timestamping as well as predictable pattern readout, avoiding the issues discussed previously for DVS/DAVIS, by making the scanning-pattern readout a design choice. The internal timestamp is then applied to one entire readout (called an 'event frame'), meaning the granularity of timestamps is much lower, normally in the order of 100s of microseconds. The DVXplorer (Lite) cameras ignore this internal timestamp and add their own in the FPGA readout logic, to be compatible with multi-camera sync features. The DVXplorer Mini / Micro cameras instead output the original timestamp and data as generated by the sensor internals (as there is no FPGA between sensor and USB output).

9.3.9 Can events be lost?

Regarding loss of events, there's fundamentally three places this can happen:

1. inside the chip itself, as explained in the *previous quesiton*, if the readout bus is overwhelmed, events are waiting to be read-out, and new events that should exist due to new changes in light would simply not happen. There is no way to 'solve' this or even understand if it did truly happen or not in the current chips, that's just how the chip and the readout mechanism are designed.
2. inside the FPGA logic (not applicable to all devices, notably not DVXplorer Mini / Micro), events are sent out via USB, but it can happen that, for a variety of reasons (host being busy, bad drivers, etc.), the USB data transfer stalls. We try to make this as unlikely as possible and use buffers as big as possible on the device side, but in the end USB is a host-driven protocol, and if the system decides that now is not the time for USB data transfers, there's nothing we can do. So once the buffers on the device side fill up, you loose events. You never loose timing information, as we do safeguard against that in our protocol. In general this kind of failure is relatively easy to spot manually, as if you plot your event stream over time, this shows up as a hole with no events at all, of any type.
3. on the host side, in our software, parsed data is put on a buffer of fixed maximum size. If the user application part that consumes data packets is much slower than the rate at which we create new ones, due to heavy processing for example, the packets queue up and at some point get dropped. We try to notify the user if possible of this happening.

NOTE: loss of data due to USB itself can't happen (in a correctly working system). We use USB bulk transfers[168] to move the data, so USB does the error detection, correction and retransmission for us. The moment we send out data via USB, when the device tells us it's fine to do so, it will arrive on the host and it will be correct and our application will get it. The only way that could fail is if there is a hardware error in that path (bad cable, bad motherboard, etc.), or a grave software error at the kernel/driver/host USB controller firmware level. Hardware-related failure usually shows up as not

[168] https://beyondlogic.org/usbnutshell/usb4.shtml#Bulk

seeing the device, or it randomly disconnecting, or the data just being corrupted (which we do detect and log), mostly due to bad cables or USB ports. Software-related failure, at that level, would be difficult to detect, as the very system you rely on (kernel/drivers) is misbehaving. We fundamentally rely on the USB stack to be working well, and we're not aware of any way to detect failures at that level in a reliable and cross-platform manner.

CHAPTER

TEN

SUPPORT

10.1 Public Forums

We maintain multiple forums that are community-driven.

- Software Support Forum[169], mostly for issues related to software usage of our cameras.
- Devices Support Forum[170], mostly for hardware issues with our cameras.

We usually advise you to look through these forums first to check if your issue hasn't been encountered and solved before contacting us.

10.2 Personalized Support

- In case you have an issue or a specific request, hardware or software-related, please reach out to us at support@iniVation.com

[169] https://groups.google.com/g/dv-users
[170] https://groups.google.com/g/davis-users

CHAPTER

ELEVEN

OTHER RESOURCES

11.1 Videos

- Some tutorial videos[171] are available on how to use *DV Modules*.
- More introductory videos on various topics[172] about neuromorphic and event-based vision are available on our website.

11.2 External Resources

- Robotics and Perception Group (RPG), University of Zurich (Web Page[173]).
- RPG's Event-based Vision Resources[174]

[171] https://inivation.com/developer/tutorials/
[172] https://inivation.com/developer/videos/
[173] https://rpg.ifi.uzh.ch/
[174] https://github.com/uzh-rpg/event-based_vision_resources