LSD Analyzer

Documentation

(by Vladyslava Fantova, Tomáš Čeleda)

(Supervised by: Jan Schäfer, Rüdiger Foest)

(Garant: Vítězslav Kříha)

2012

1 Setup

The apparatus consists of the laser (1) (HeNe Linos power supply PS 1235, 633 nm, 5 mW) and high speed camera GigE DR1-D1213-200-G2 (2), which is placed into the laser path 470 mm from the plasma jet. The lenses are used to decrease the necessary distance between the laser and CMOS camera to get larger deflection (see [1]). The 20% grey filter is necessary to decrease laser beam intensity hitting the camera chip. Camera is used without any optics.

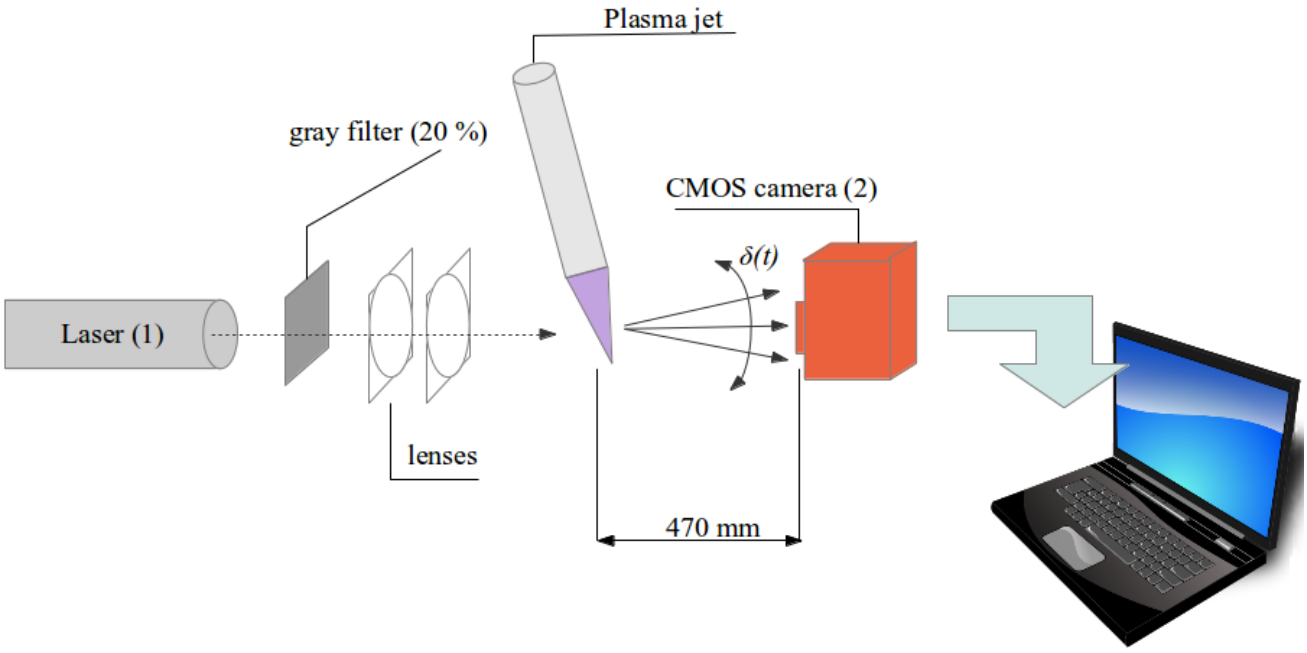
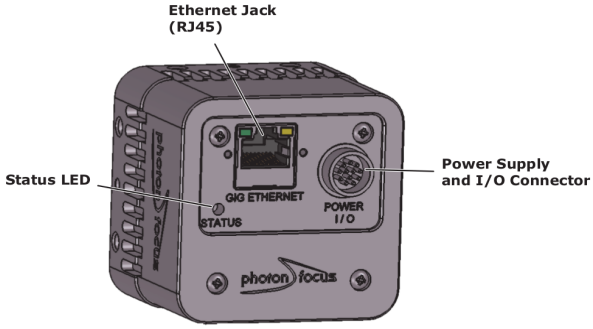


Fig 1. The setup of the experiment. Plasma jet is set to the z-axis.

1.1 High speed camera **GigE DR1-D1213-200-G2**



- Interface: Gigabit Ethernet
- Greyscale resolution: 12 bit / 10 bit / 8 bit
- Technology: CMOS active pixel (APS)
- Scanning system: progressive scan
- Resolution: 1312 x 1082 pixels
- Pixel size: 8 µm x 8 µm
- Exposure time: 10 µ s … 1.67 s
- Exposure time increment: 100 ns
- Pixel clock frequency: 40 MHz
- Pixel clock cycle: 25 ns
- Operating temperature: 0°C – 50°C
- Power supply: +12 V DC … +24 V DC
- Dimension: 60 x 60 x 51 mm^3

All parameters, installation guide and other information is specified in the camera user guide.

The camera is already installed on the laptop and it is not necessary to change any settings, that are not specified in present documentation.

To control weather camera is on check Ethernet LEDs. The Status LED orange color indicates any problem both with communication and hardware (see camera user guide).

1.2 Laser Linos

- Wavelength: 632.8 nm
- Output power: 5 mW
- Working gasses: He Ne
- Beam diameter: 0.49 mm
- Divergence: 1.7 mrad
- Polarization: N/A

1.3 Notebook parameters and connection

The camera is connected to notebook Dell Latitude E6510 with 1 Gb/s Ethernet card. The speed of the card was sufficient; however the card doesn't support so-called jumbo frames recommended for large images transfer by the camera manufacturer.

Some laptop parameters, that are necessary for the present apparatus setup:

- 1 Gb/s Ethernet card
- Processor: i5
- RAM: 4 GB
- SSD disk transfer rate: 135 MB/s

2 Experiment concept

Two modes of measurement were discussed:

- 1D mode
  - Used image size: 544 x 1 px
  - Maximal possible frame rate: 10 000 fps
  - Data loss: minimal, usually no.
  - Disadvantage: x-axis movement detection only
- 2D mode
  - Image size: 544 x 200 px
  - Maximal possible frame rate: 800 fps
  - Data loss: cca 1 %
  - X and y axis movement detection

Both modes are implemented in the LSD Analyzer.

The 1D mode is recommended for normal measurement. The 2D mode should be modified to decrease data loss.

3 Software

3.1 Software usage

Run the program

The source of LSD Analyzer executable is located in
C:\Users\Jan\Documents\Visual Studio 2010\Projects\LSDAnalyzer_1D_final\Debug\LSDAnalyzer.exe

Or you can open LSD Analyzer project in MS Visual C++ Express by opening file LSDAnalyzer.sln. Press the Run button (F5 key) to compile and start the program.

Program will ask you to enter the experiment name which will appear in output file name. Type it, press enter, connect to camera via dialog (Fig 2) and wait. Output file destination can be set in configuration file, default location is E:\Lab. The directory must exist before program startup.

It is not recommended to work with PC while the program is running. Try not to touch keyboard or mouse.

Watch the console window for possible errors during the data capture.

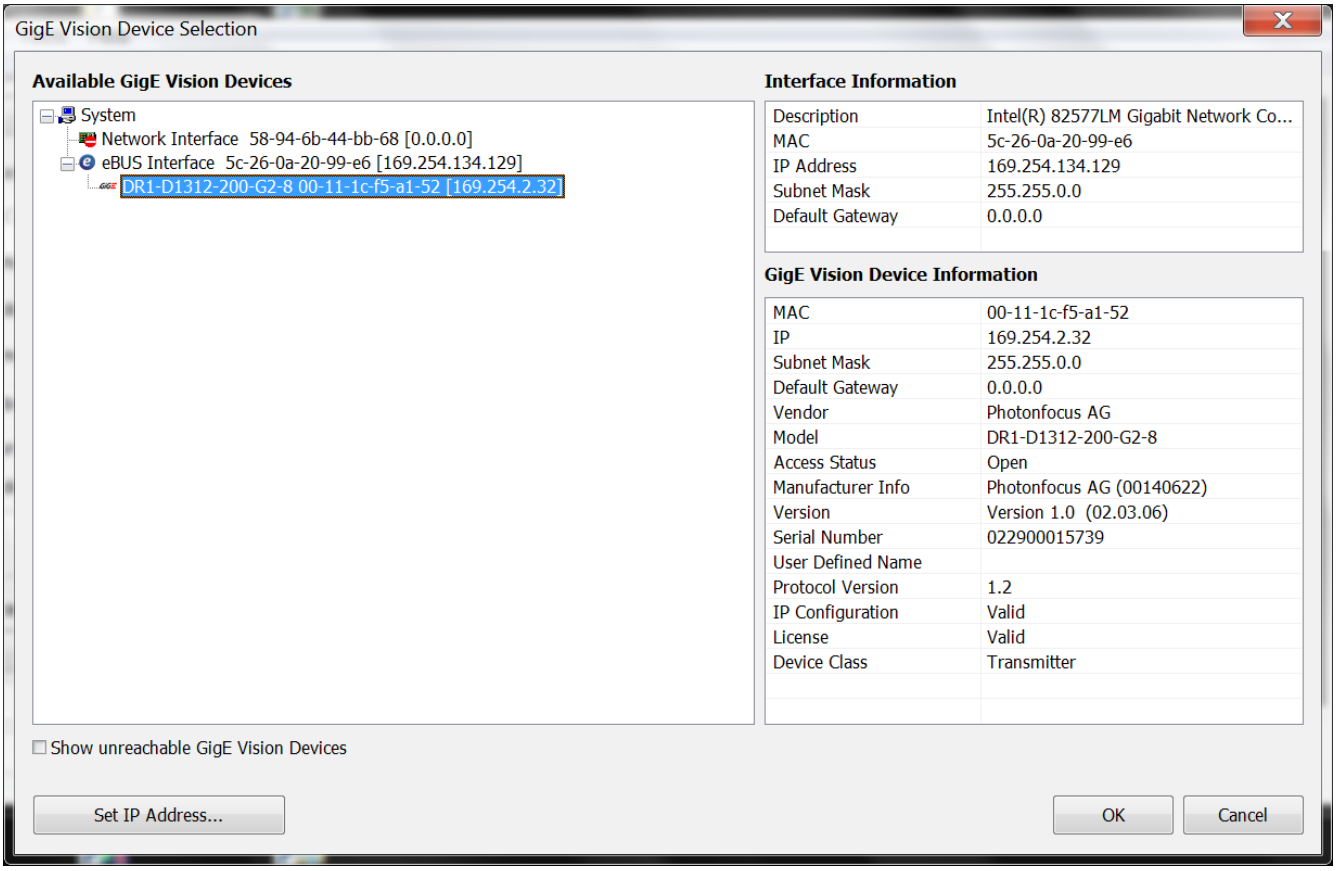


Fig 2. Camera connection dialog (camera can be connected within one program)

Autocalibration

Before the first data recording you should calibrate the camera. To start the calibration type “cal” instead of the name of experiment. After the calibration is done, optimal values of gain, x and y image offset are written to the terminal and stored to the camera memory. Press enter to close the window and start program again to measure data.

3.2 Configuring parameters

Configuration parameters

There are three group of camera parameters:

1. Parameters, that are set through the **configuration file** (see part Configuration file). These parameters are set by LSD Analyzer to the camera each time the measurement is started.
2. Parameters, that are calculated within the **autocalibration** and are set by the LSD Analyzer after the calibration is finished. Parameters are stored in camera memory until the camera is switched off.
3. Parameters, that are set by the **PF Gev software**. (see fig 3, 4). These parameters are stored in camera memory until the camera is switched off. It is typically no need to change these parameters. Detailed description of all parameters are in PF Gev documentation.

Via configuration file

Configuration file location: File is located in the same directory as LSDAnalyzer.exe and is named *settings.ini.*

OutputDataPath – the path and the name of the file, where the read data will be saved. The experiment name set by the user, will be used as a prefix.
Default value is: E:\Lab\%s_DATA.txt

ErrorPath - destination of the error output file. Data loss and brightness under threshold errors are written into the log file.
Default value is: E:\Lab\%s_ERR.txt

ExperimentDuration – set the experiment length in seconds. Max value: 30 min;
Default value is: 10 s

CameraFramerate – set the camera framerate. Maximal recommended value for 1D image is 10 000 fps.
Default value is: 10 000 fps

BrightnessThresholdValue – set the threshold to calculate x and y laser spot center. Value can variable from 0 to 255. Recommended value range is 150 to 190.
Default value is: 180

CameraExposureTime – set the the time of camera exposure in microseconds. Range: 10 µs .. 1.67 s.
Default value is: 10 µs

CameraBlacklevel – set the camera black level. Optimal value is 100. Range is 0 .. 255
Default value is: 100

OptimumMaxBrightness – the parameter for gain penalty function used in the autocalibration. Range of this parameter is 0 to 255, but it is recommended to set it in range 220 .. 240.
Default value is: 230

UdpBridgeEnabled – the parameter to enable the UDP bridge to convert data to the binary header file. To enable set it to 1.
Default value is: 0 (disabled).

DastinationIpAddress – the parameter, where the IP address of the destination computer is set. It is no need to set it, if the UDP bridge is disabled.
Default value is: none

DestinationUdpPort – the number, that defines the destination UDP port of UDP bridge.
Default value is: 666

Setting parameters with autocalibration

Autocalibration procedure finds optimum gain, x-offset and y-offset of the camera. These parameters can differ when camera is touched. The procedure tries a range of values for each parameter to find an optimum value. Optimum value is a minimum of a penalty function that is defined for each parameter separately.

Gain penalty function is defined as absolute value of the difference between maximal brightness of the image and optimal gain parameter, that is defined in the configuration file.

Offset X penalty function is defined as distance of computed laser beam spot center from the half value of image width.

Offset Y penalty function is defined as negative count of pixels which brightness is above BrightnessThresholdValue, that can be set in the configuration file.

Via PF Gev

The program PF Gev offers the possibility to see the camera image in real time and set all device parameters.

Before you connect the camera (with te Select/Connect button) don't forget to check, if the camera is not connected within any other software. You should disconnect the camera in PF Gev before using of LSD Analyzer. To see the real time camera image press Play button. The image will appear in the right dialog. It is possible to set x and y offset in real time.

To set the device parameters, use the Device control button. It is possible to set all parameters, that are set by LSD Analyzer, but keep in mind, that LSD Analyzer will overwrite values, that are set in its configuration file. And it will overwrite values of gain, x and y offset as soon as the autocalibration will be started.

LSD Analyzes uses a single row of data (height value is set to 1px). To control autocalibrated parameters, run the autocalibration within LSD Analyzer, wait until camera is disconnected and connect it with PF Gev. Move camera if the spot is not situated in the center of the image.

The row image can be seen only if the PF Gev is in full screen mode (the window must be large enough).

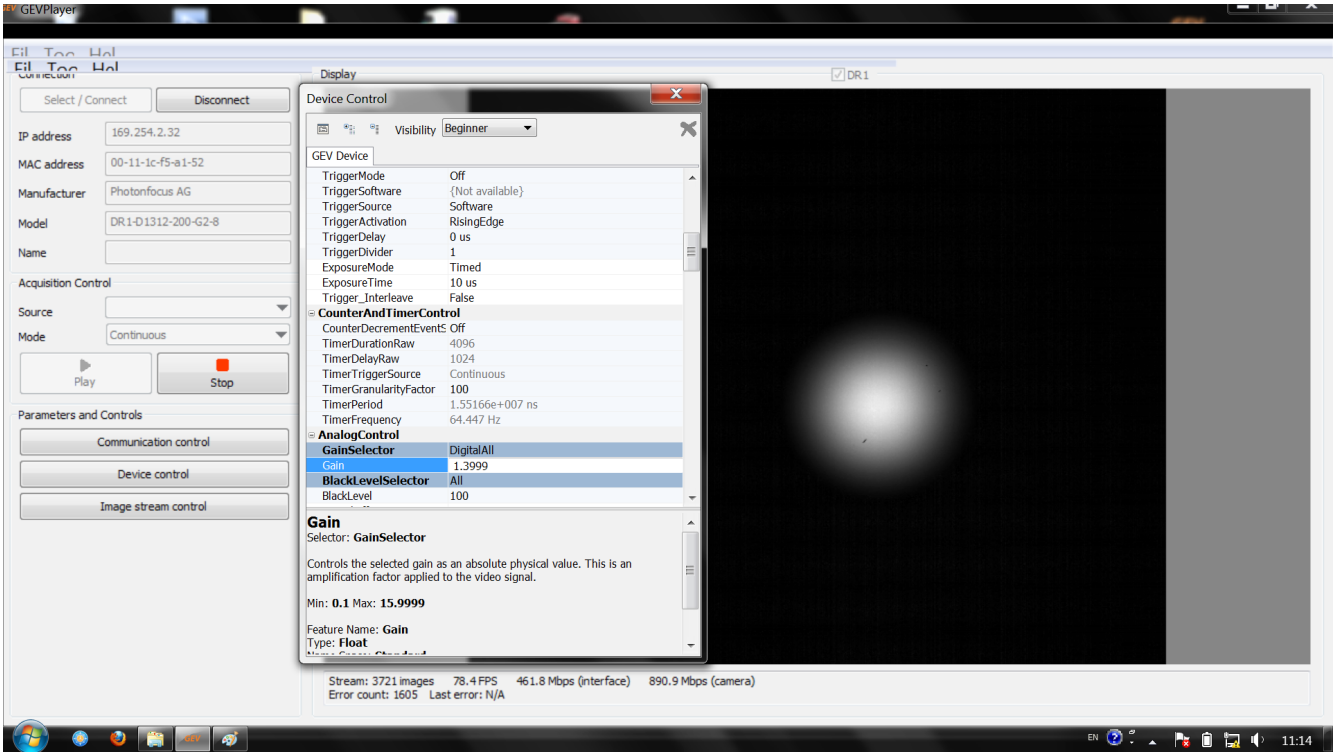


Fig 3. PF Gev offers the possibility to see the camera image in real time and set all device parameters

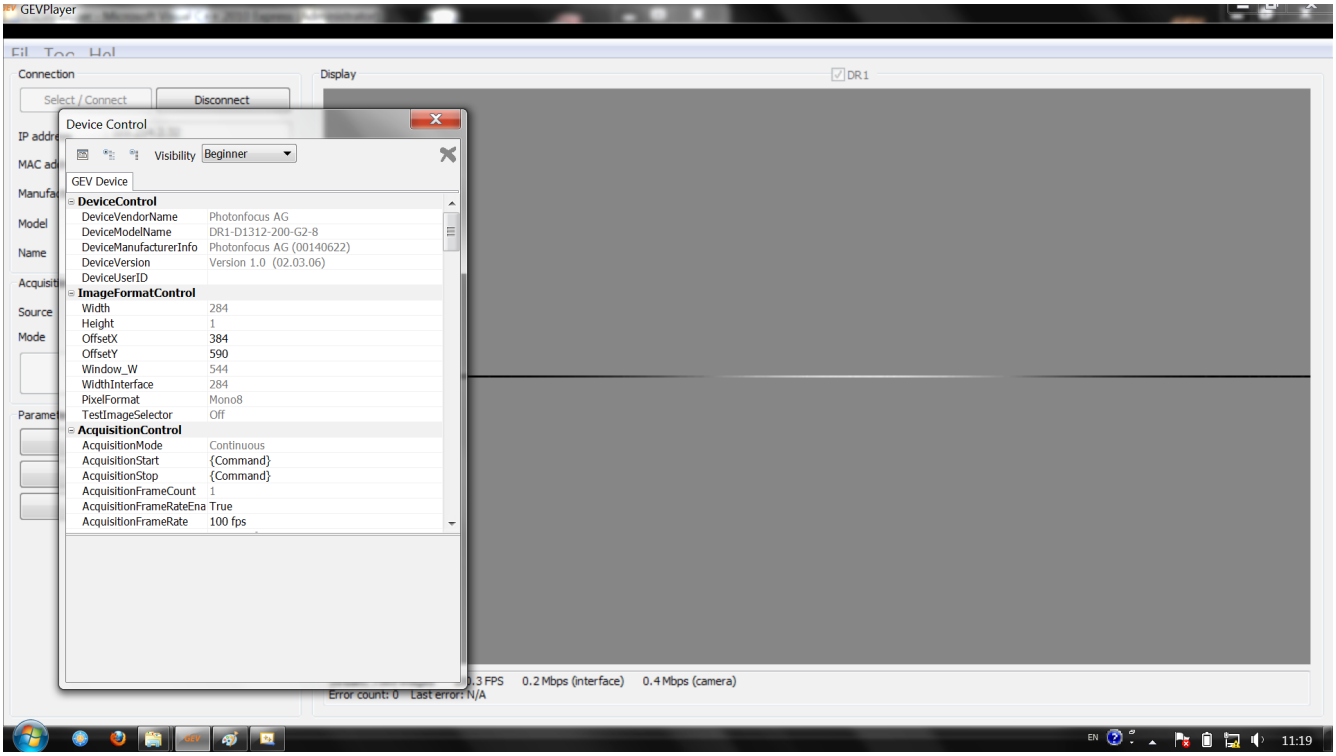


Fig 4. One row image with parameters set by the LSD Analyzer.

3.3 Software output

The **output** CSV file of LSD Analyzer contains these columns:

- Image number – index of captured image
- Block ID – this value is taken from the camera and it's maximal value is 65 535. It is used in the program to check weather blocks aren't lost.
- Frame rate – is the actual framerate. The variability of the framerate is very little, so it shouldn't make a big error to use its mean value during the FFT calculation.
- TimeStamp – is taken by the camera. See camera documentation for more information.
- X – is the x-value of the calculated center of the laser spot on the camera chip. It is calculated as the mean of x-coordinates of pixels which brightness is over threshold. The threshold is set by BrightnessThresholdValue parameter.
- Sx – is the standard deviation of the pixels over threshold for x-axis
- Y – is the x-value of the calculated center of the laser spot on the camera chip and is calculated in the same way as X one. If the height of the image is set to 1px the Y is 0.
- Sy – is the standard deviation for Y-axis
- Count – the number of pixels which brightness is higher then BrightnessThresholdValue parameter. This value can be used to control the threshold. If there are too many (over 120) or too little (less then 20) pixels with brightness over threshold, the threshold value should be changed. 180 pixels value of threshold was selected experimentally.

- Max – is the maximum brightness value in the image. It is used to control weather the camera gain is set correctly and weather the image is not overexposed.

3.4 Implementation details

Application uses Windows SDK to capture frames from the camera.

The implementation is focused on maximizing performance of reading data from camera. Camera sends data via Ethernet connection to computer where camera control library stores the data into buffer. The buffer size is limited to only 64 frames. Camera framerate can be set up to 10 000 frames per second. If application didn't process the data in input buffer fast enough, some of the frames would be lost.

Application uses two threads to capture and process frames from the camera. The first one - *capturing thread* is fully dedicated to reading data from the camera with minimum overhead possible. The second one - *processing thread* processes captured frames, calculates beam center from each of them and saves data into CSV file.

Communication between the two threads is maintained via one buffer and one global shared variable. In order to maximize performance of capturing thread, there is no standard synchronization mechanism like mutex or semaphore. This mechanism would cause occasional thread blocking and thread rescheduling that would lead to decreased performance.

Instead we have used a shared variable as thread synchronization. The capturing thread is incrementing a counter variable for each frame stored into cyclical frame buffer. While the processing thread is constantly polling that variable checking if it has changed. When the variable changes, the processing thread processes the appropriate number of frames. Only the capturing thread is incrementing the variable and the processing thread is reading it. This mechanism is race condition safe since writing into an integer variable is an atomic operation.

Processing thread can process frames by a sufficient margin faster than the capturing thread can capture them. Since frame buffer between the two threads is large enough, situation when the capturing thread fills the whole buffer before processing thread can process them never occurs.

Despite all our effort to maximize speed of reading data, some data losses rarely occur. This data loss happens somewhere inside camera control library before the capturer thread can handle them. They are probably caused by some resource intensive tasks running in parallel. These can be some background tasks running by operating system. It is not recommended to work on computer during measurement.

UDP communication

If UdpCommunication is enabled in settings (**UdpBridgeEnabled** = 1), data about each processed frame are sent via UDP datagram.

Destination of UDP Datagram is set by **DestinationIpAddress** and **DestinationUdpPort** parameters in configuration file.

Each datagram contains information about one captured frame. Transferred data are in binary format. C language structure matching sent data is in **DataResult.h**. The data structure has layout specified in table:

Variable	Data type	Description
FrameNumber	64 bit unsigned Integer	the path and the name of the file, where the read data will be saved.
BlockId	16 bit unsigned Integer	This value is taken from the camera

FrameRate	8 Byte floating point (double)	Actual frame rate
TimeStamp	64 bit unsigned Integer	Time stamp of the image, sent by the camera
X	8 Byte floating point (double)	X axis of the calculated center of the image
Sx	8 Byte floating point (double)	Standard deviation for X
Y	8 Byte floating point (double)	Y axis of the calculated center of the image
Sy	8 Byte floating point (double)	Standard deviation for Y
Count	32 bit unsigned Integer	Number of pixels which brightness is over threshold
Max	8 bit unsigned Integer	Maximal brightness value in the image

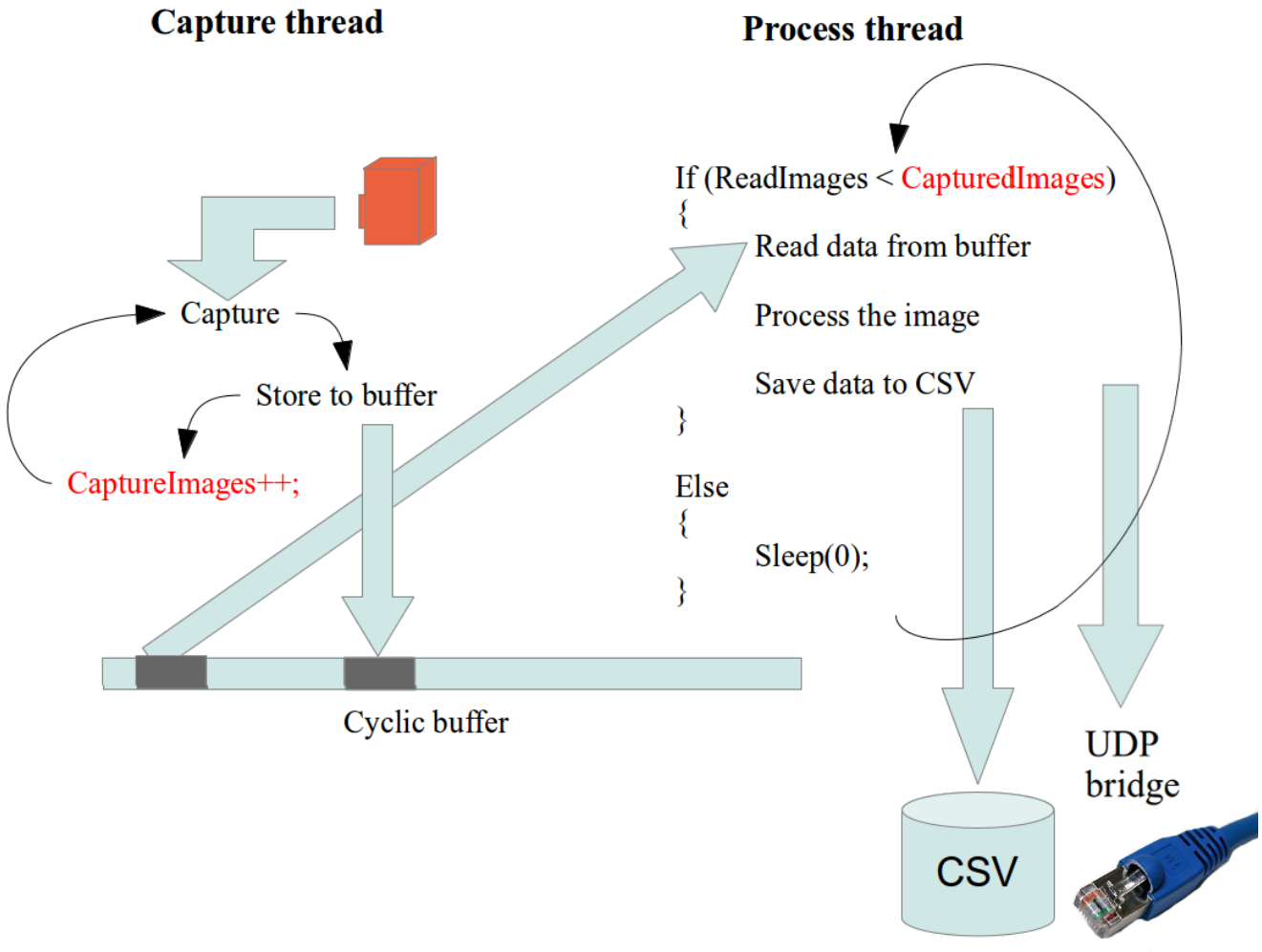


Fig 5. Thread communication diagram

4 TODOs

1. Modify the 2D mode to prevent data loss. Possible solution: disable the antivirus software and other background services run by the OS.
2. Implement **data processing**
    1. Moving average filter
    2. FFT processing and frequency detection
    3. Temperature calculation with histogram information
3. Implement the **graphical user** interface