Challenges of Using V4L2 to Capture and Process Video Sensor Images

MICROCHIP

A Leading Provider of Smart, Connected and Secure Embedded Control Solutions

SMART | CONNECTED | SECURE

Eugen Hristev

October 27, 2020

Eugen Hristev

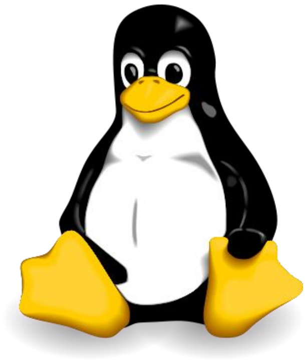
Embedded Linux Engineer at Microchip

Part of the MPU32 Linux team

Developing second stage and third stage bootloaders

Developing Linux kernel device drivers

Maintaining and developing Microchip V4L2 drivers:
Atmel sensor controller, atmel sensor interface



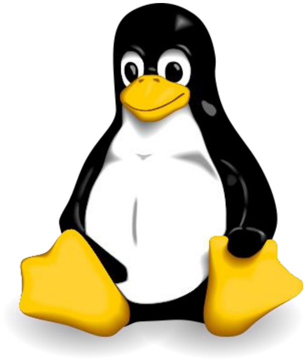
Agenda

How digital sensors work

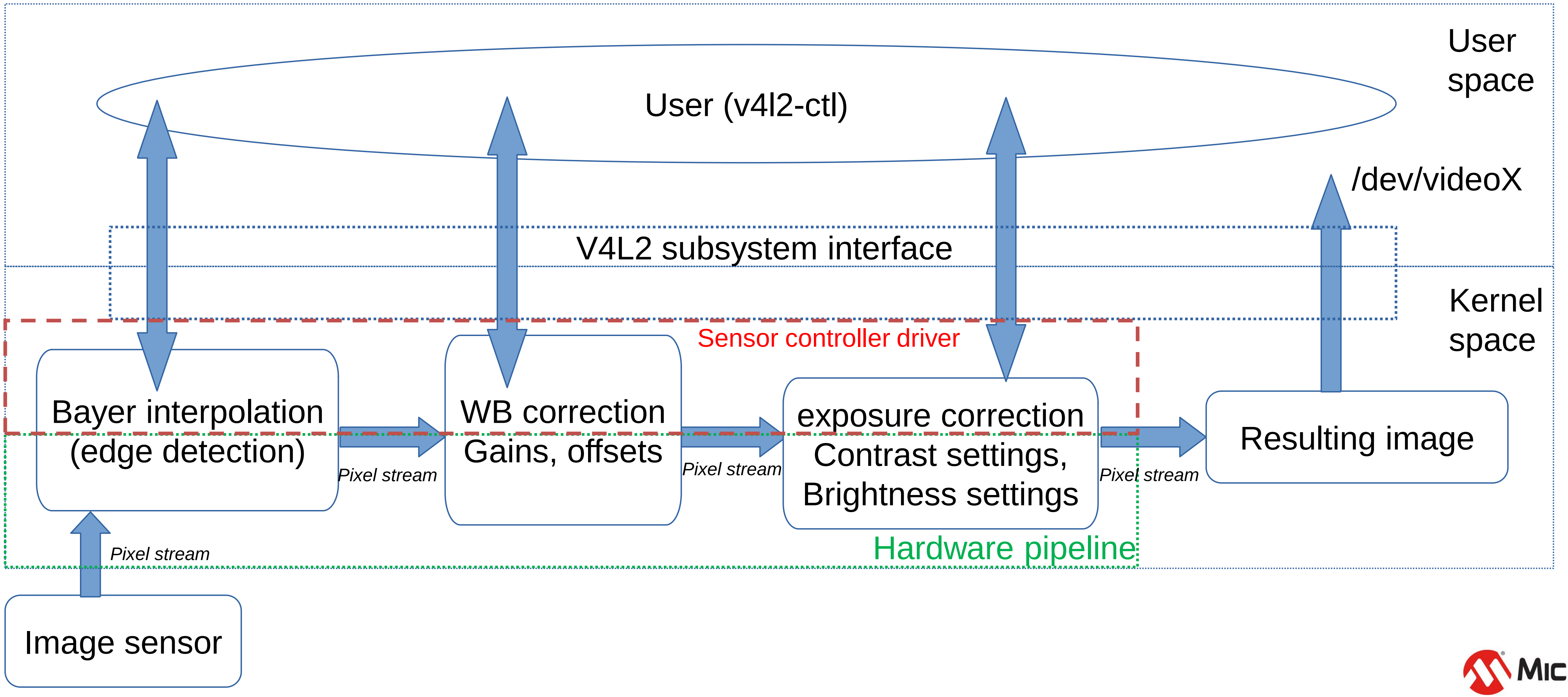
What happens with digital data until it is turned into a real photo

Issues that can occur during this process

How can V4L2 help tuning dedicated hardware and software in getting better picture quality



System diagram

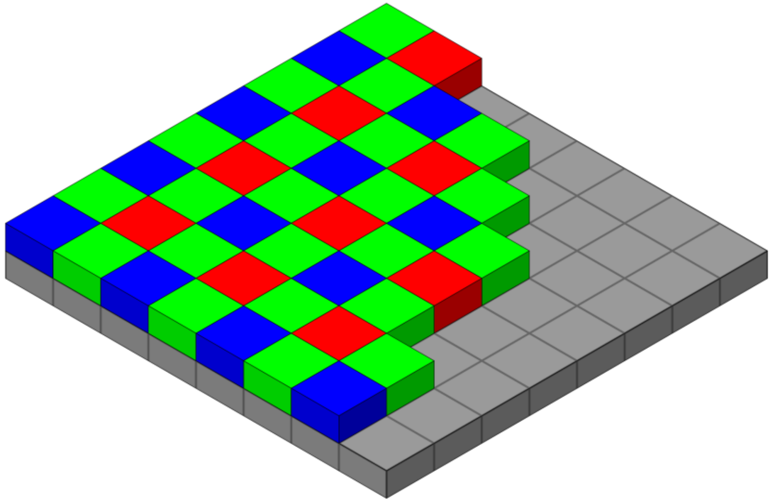


Digital video sensor

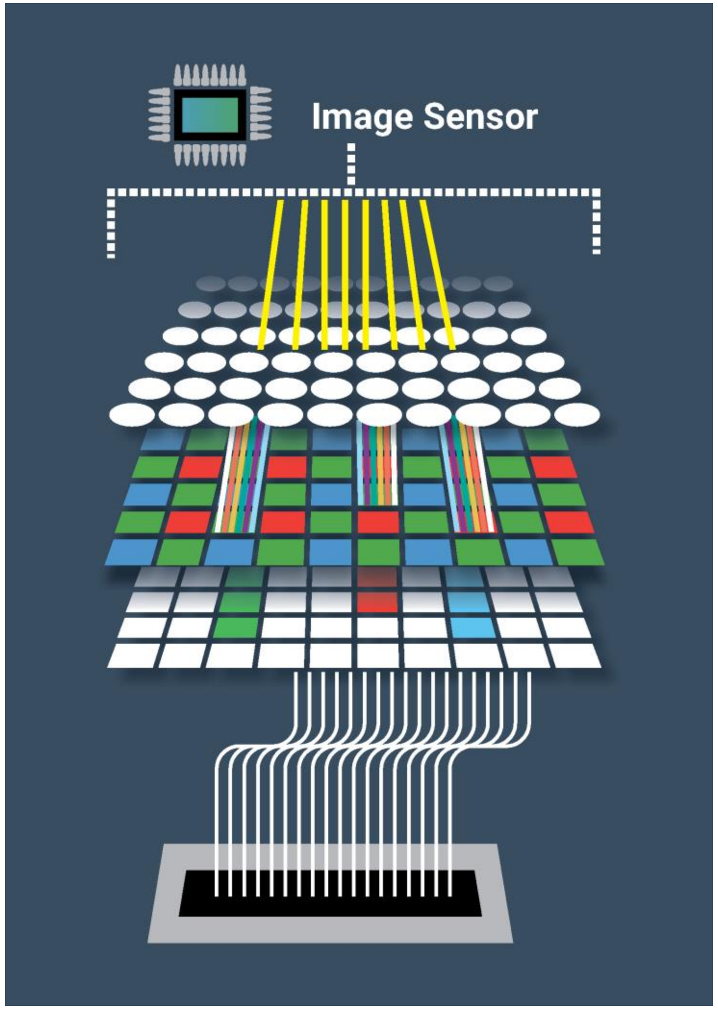
How does a sensor work ?

Capture the light

Sensors operate by capturing ambient light
Sensors have an array of light sensitive pixels



BAYER array



BAYER array

Why do we have more green pixels than red/blue pixels ?

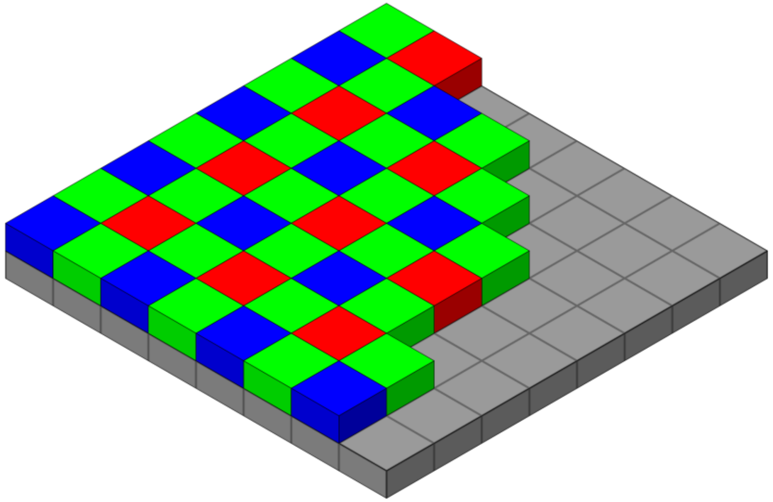
Why was this pattern chosen ?

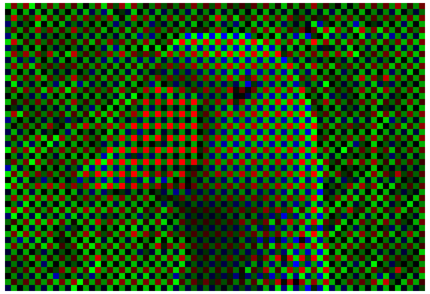
Why each pixel has only one color ?

Do we lose color information ?

How do we convert pixel color into bits ?

How do we get a real photo out of this pattern ?



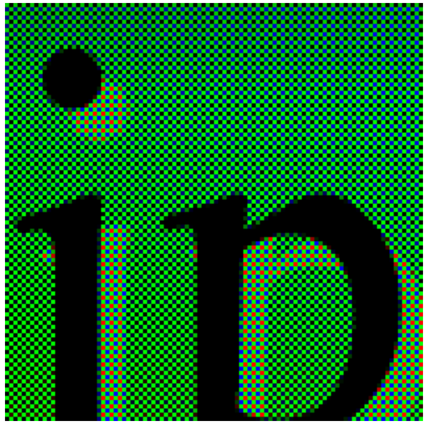


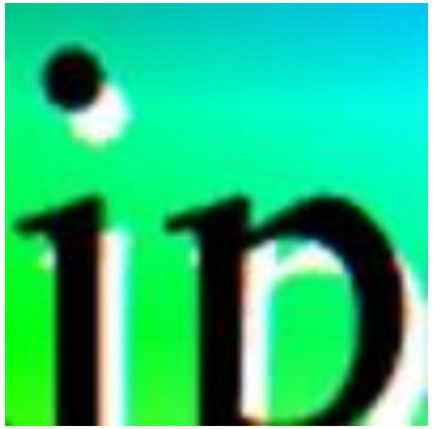
Pixel data inside controller

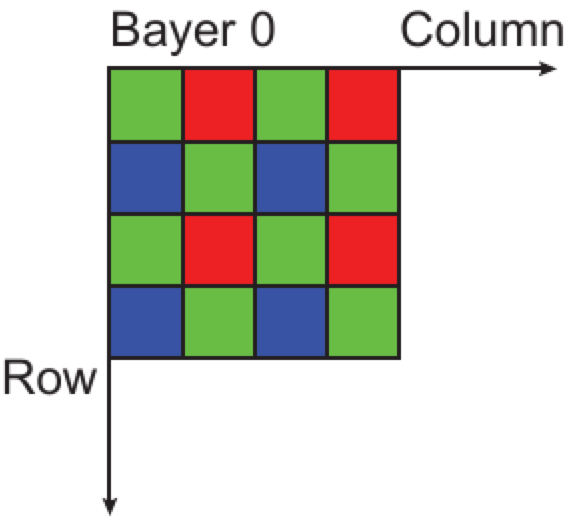
We have pixel data. What is next ?

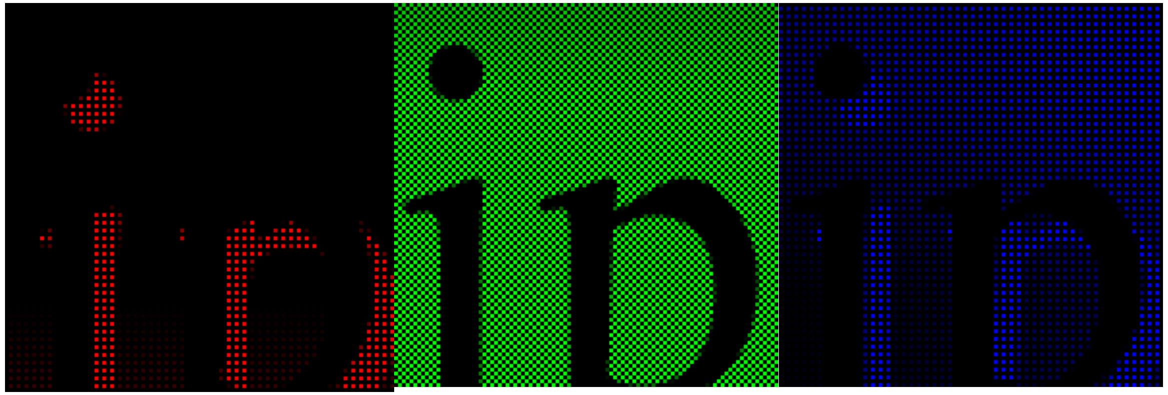
BAYER interpolation

Each pixel will obtain information from its neighbors







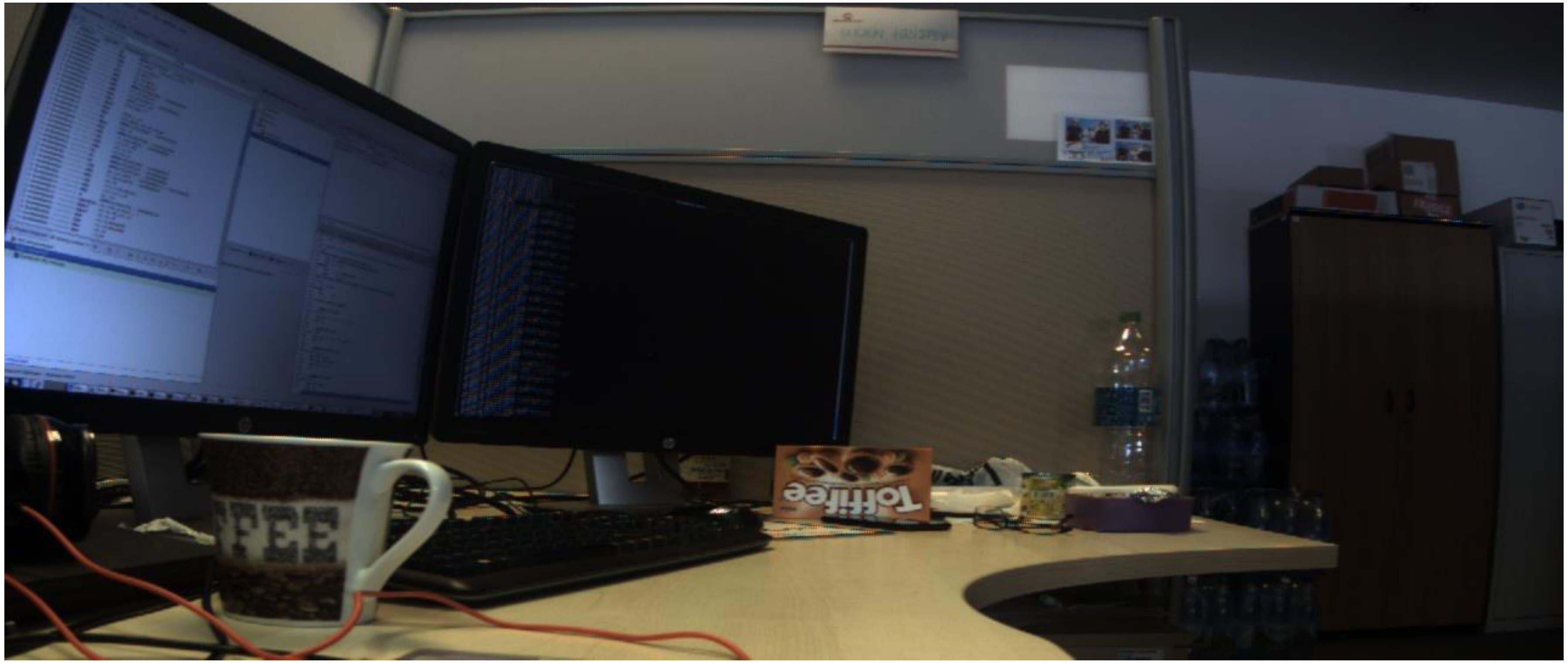


What can go wrong ?

We interpolate. But is this flawless ?

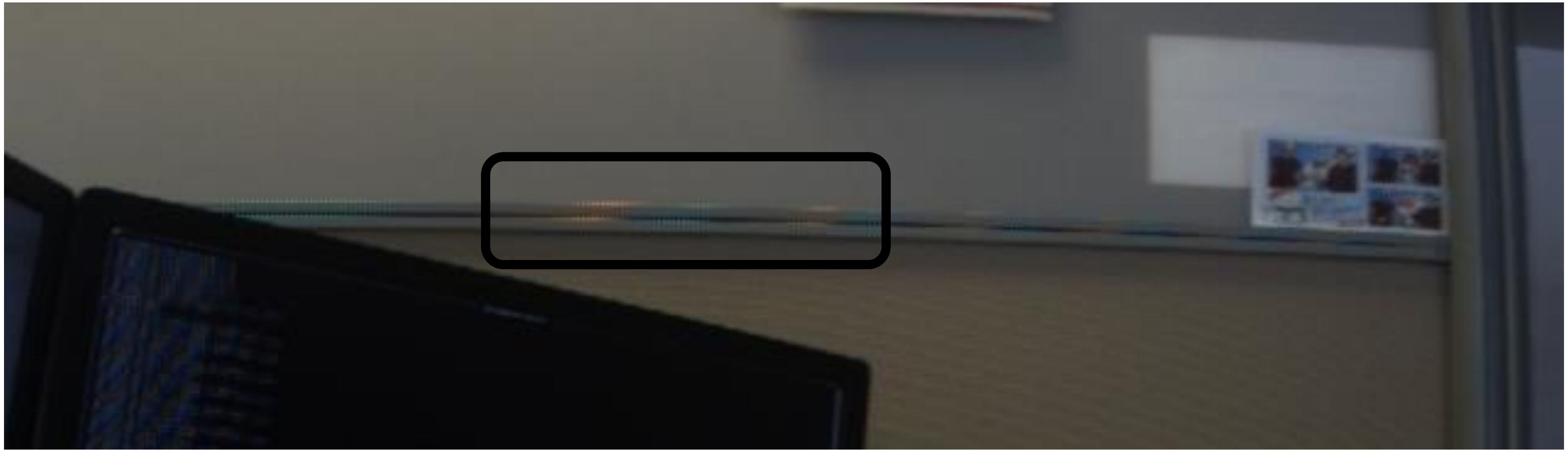
The edge problem

Pitfall of the demosaicing



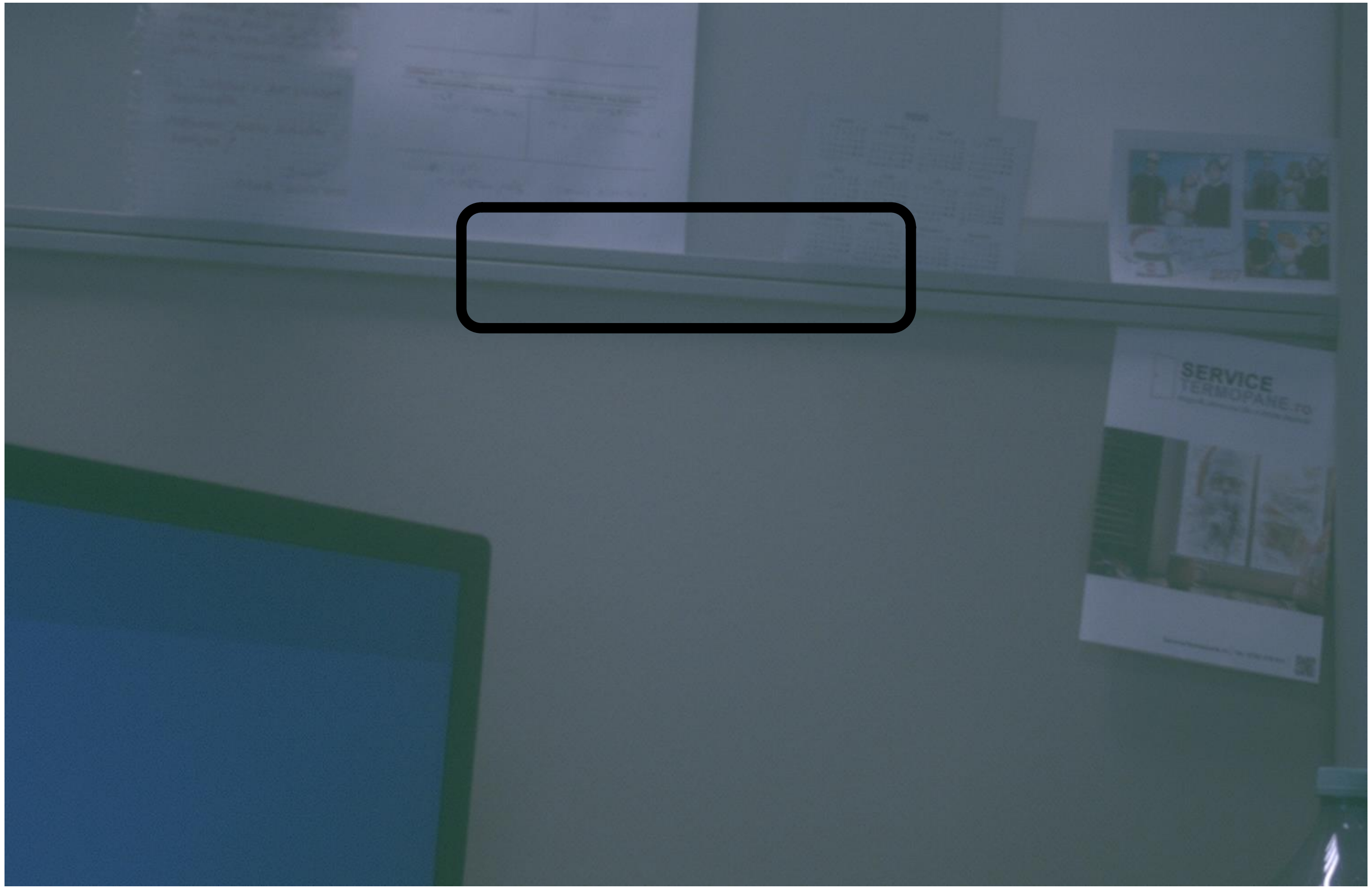
The edge problem

Pitfall of the demosaicing



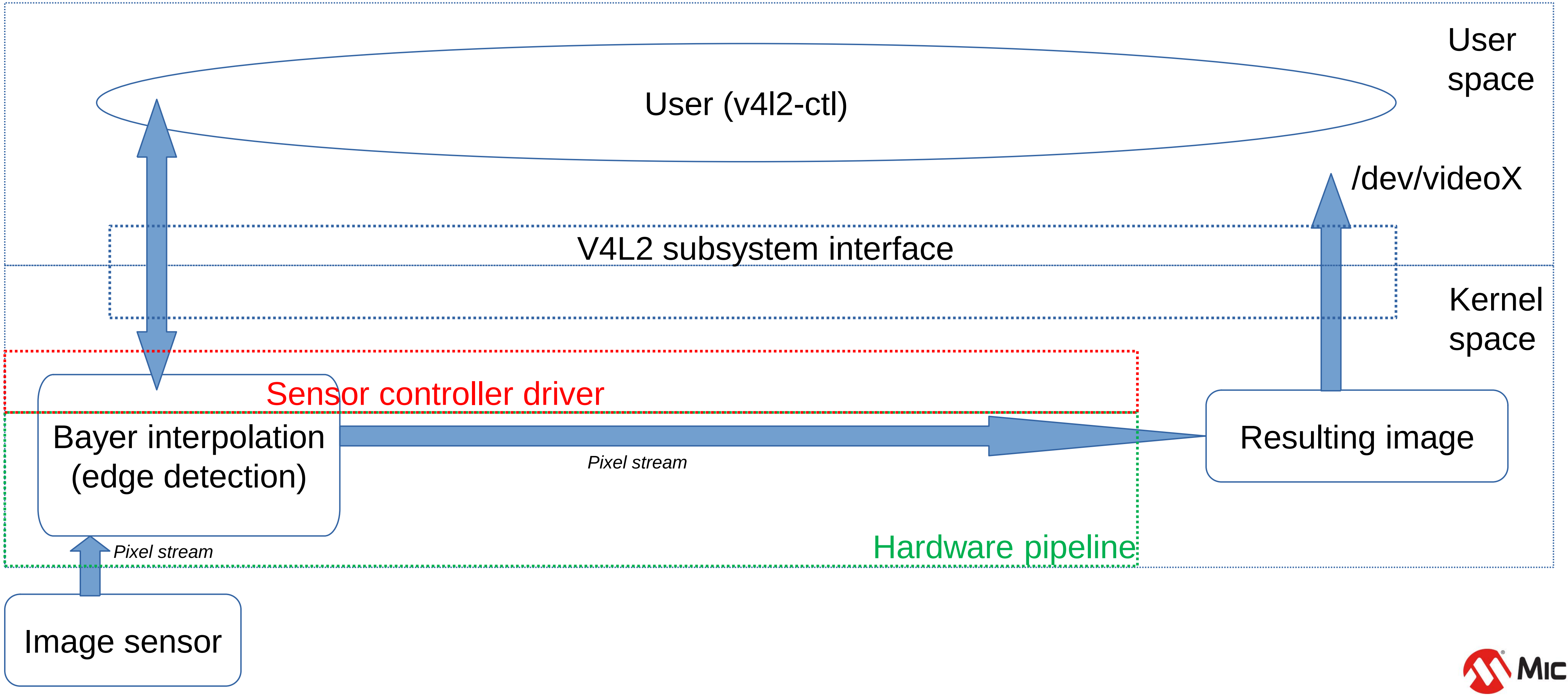
The edge problem

What can we do ?



Edge interpolation : sometimes, but not always !

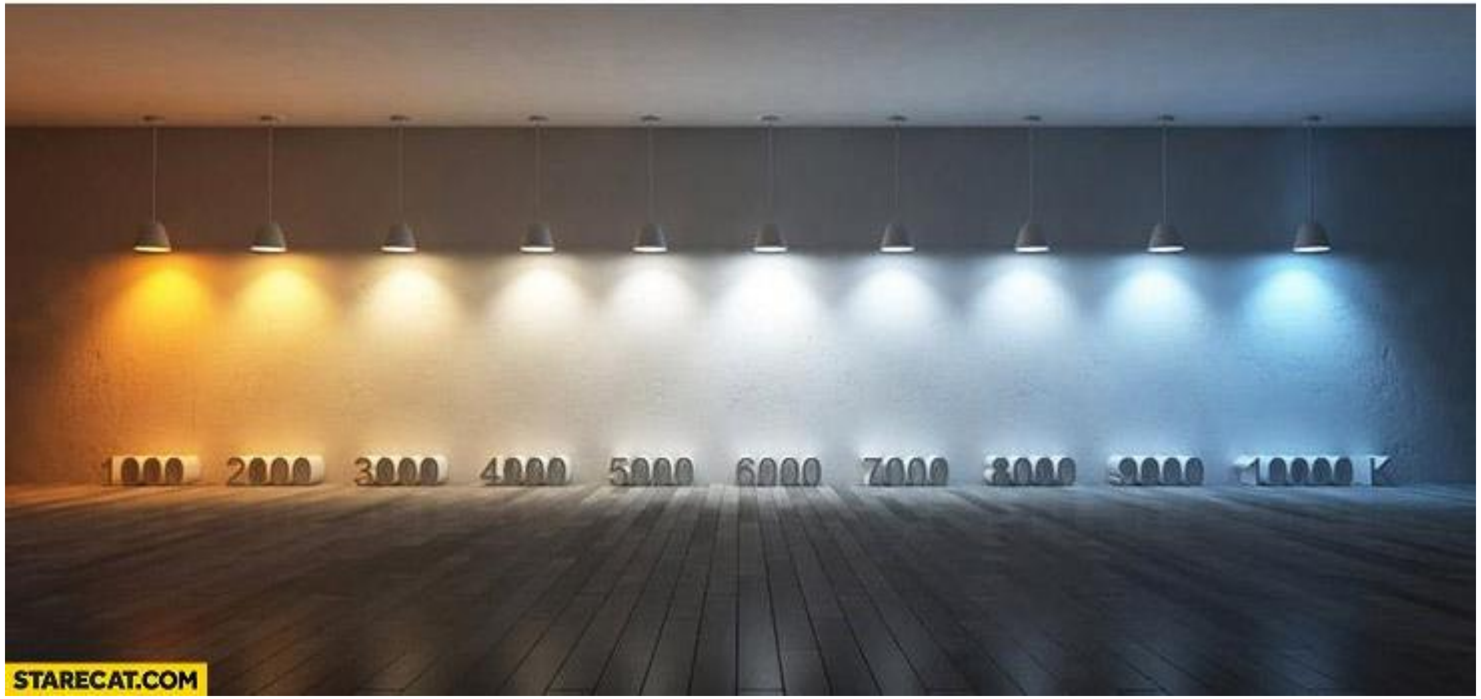
Inside the system



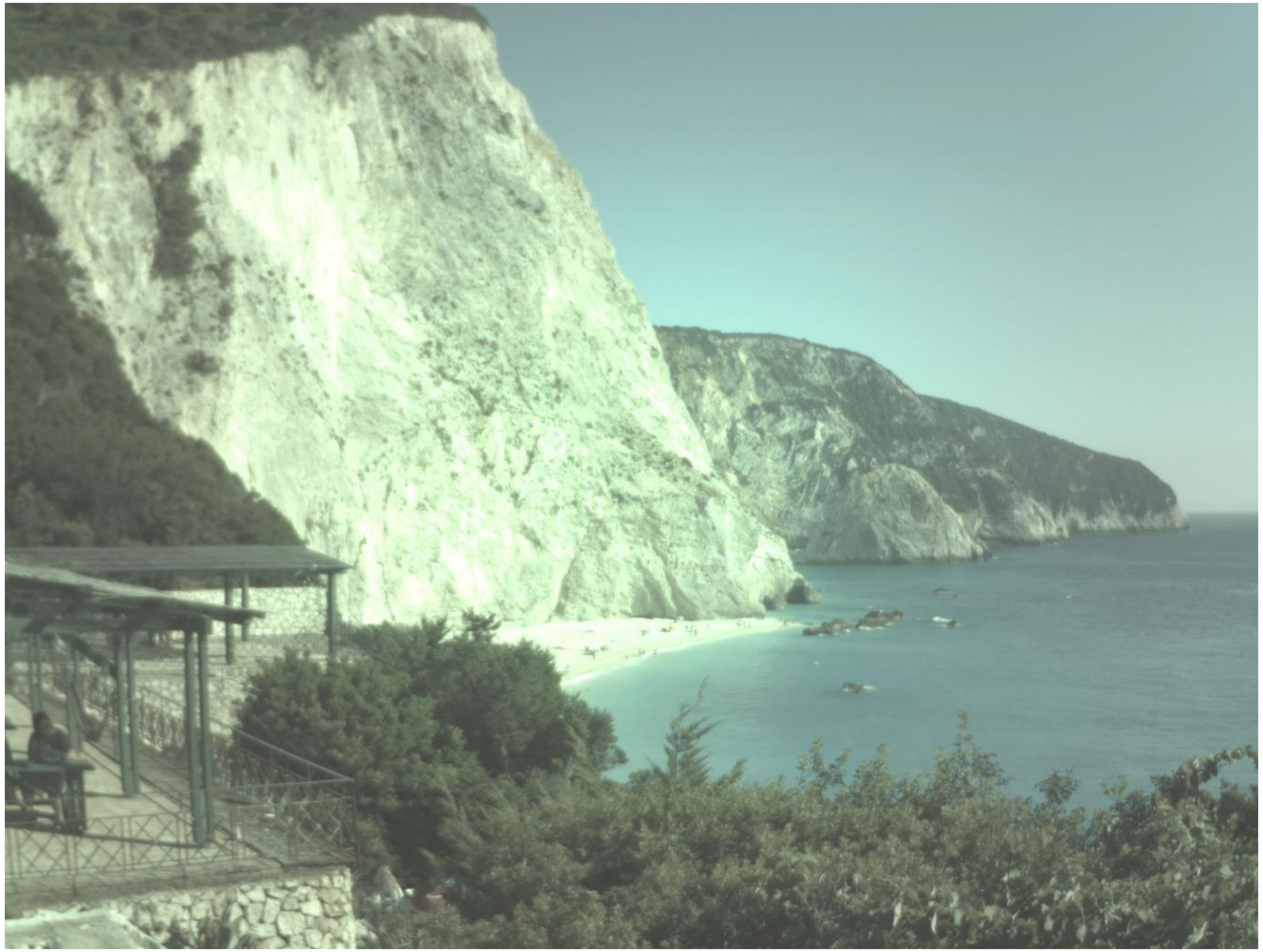
The color problem

How we see the light

Light has a temperature

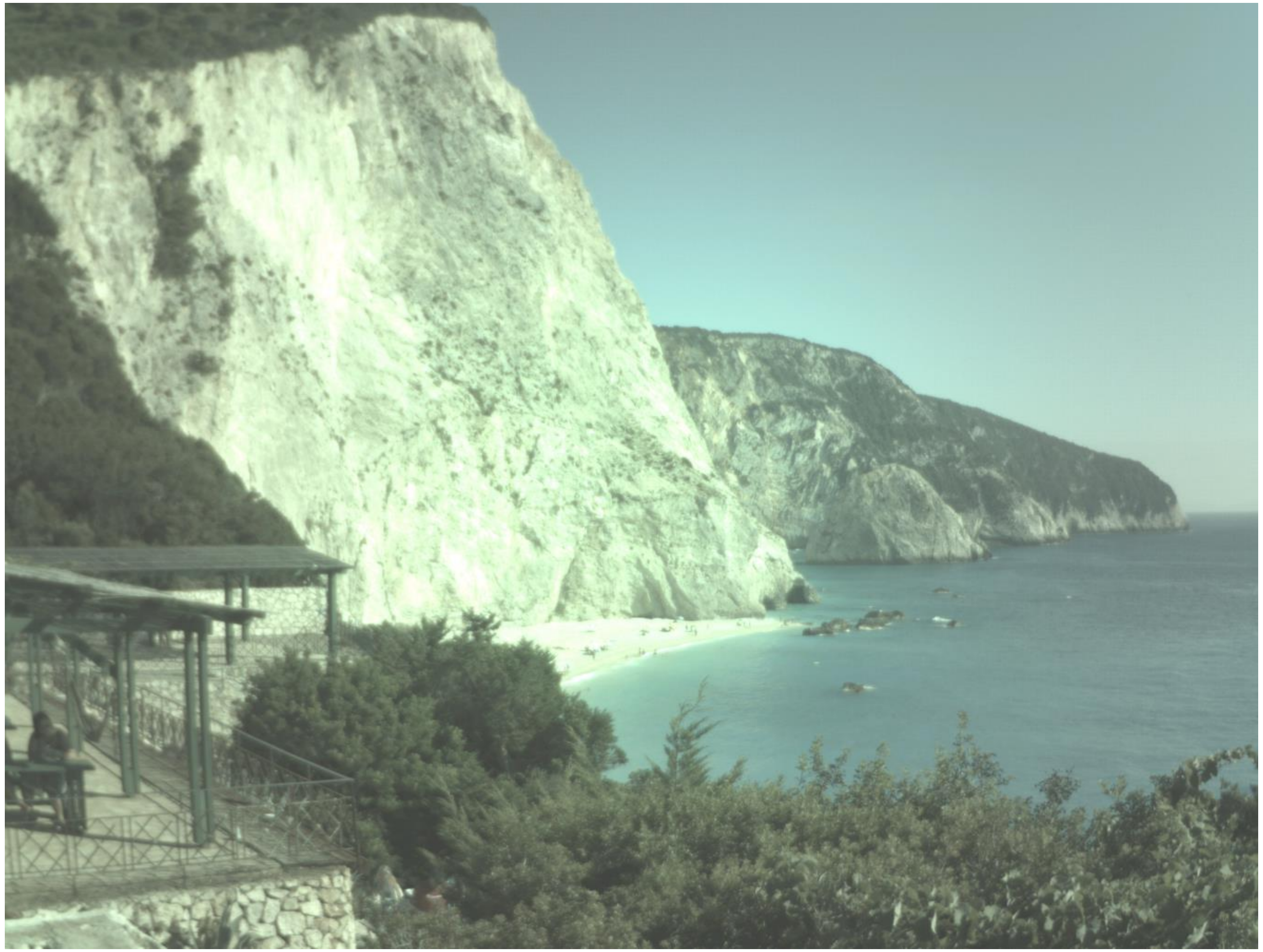


Temperature and color



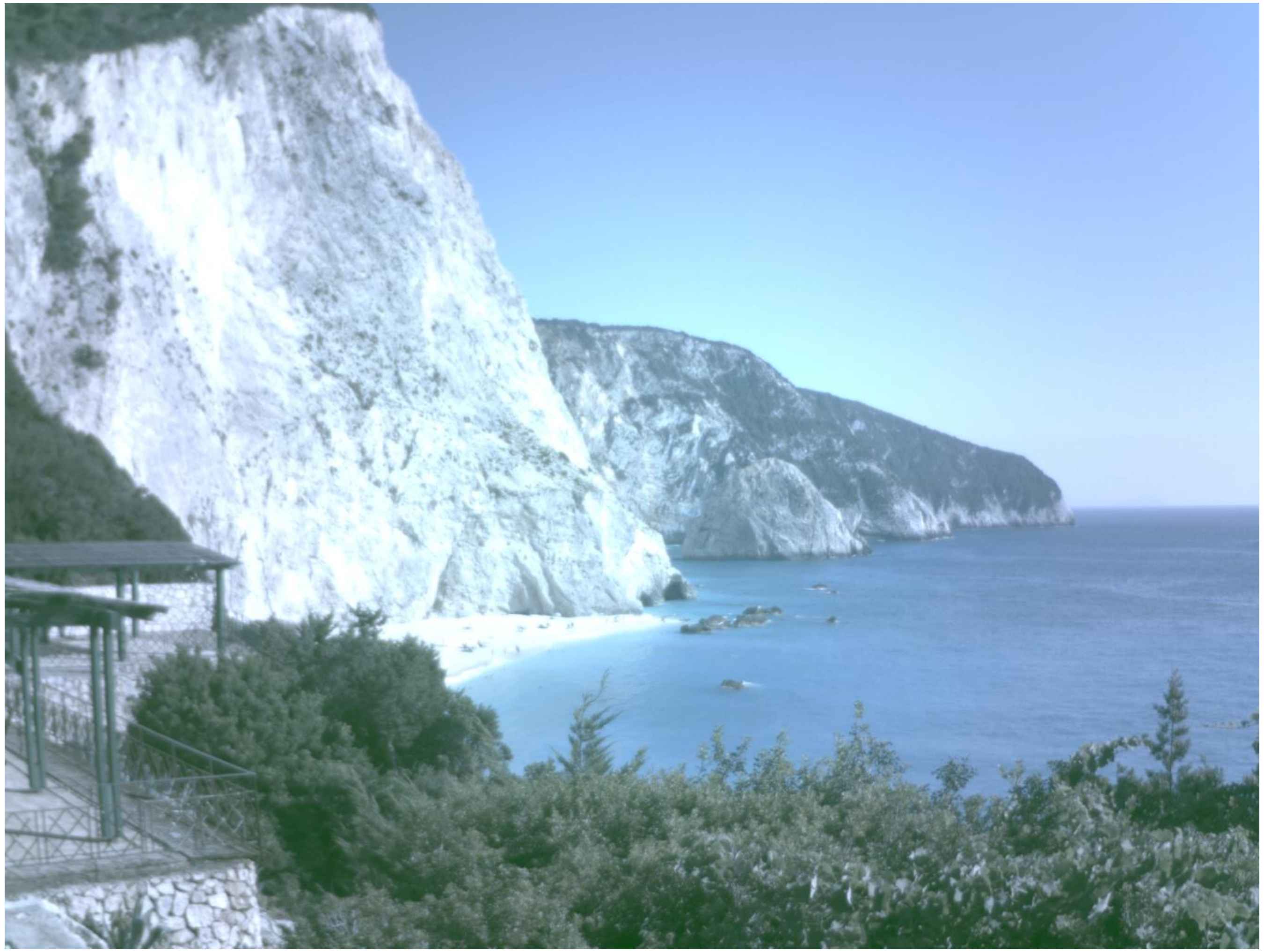
Temperature and color

What color is the water in this photo ? What about the rock ?



Temperature and color

What about now ?



What the human brain can see

Remember old black and white photos ?

Can you actually see colors in black and white ?

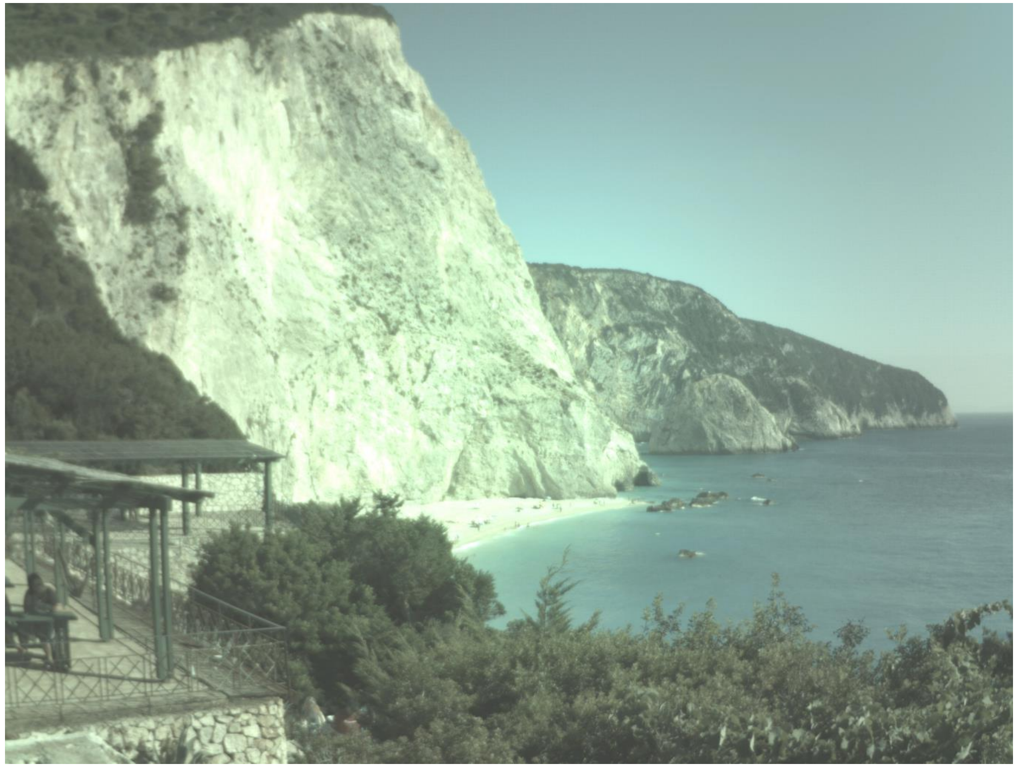
What the human brain can see

What about Sepia ?

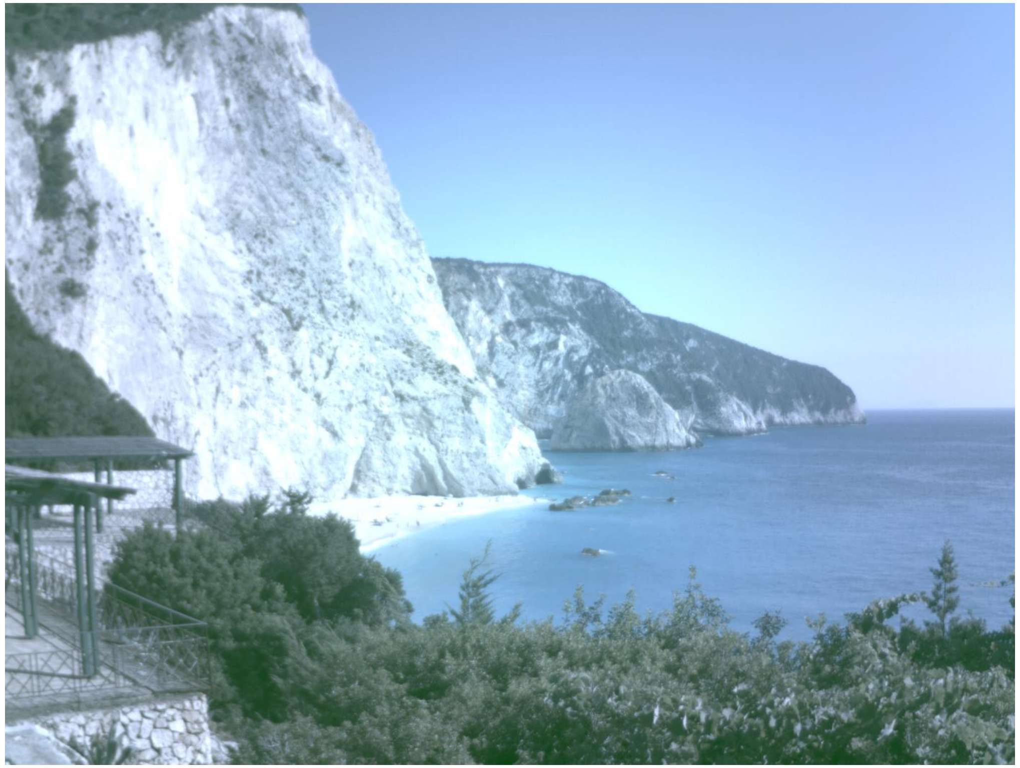
The sensor does not have a brain !

We will need to adjust the color such that they look more natural in the specific light of the scenery

What the human eye sees

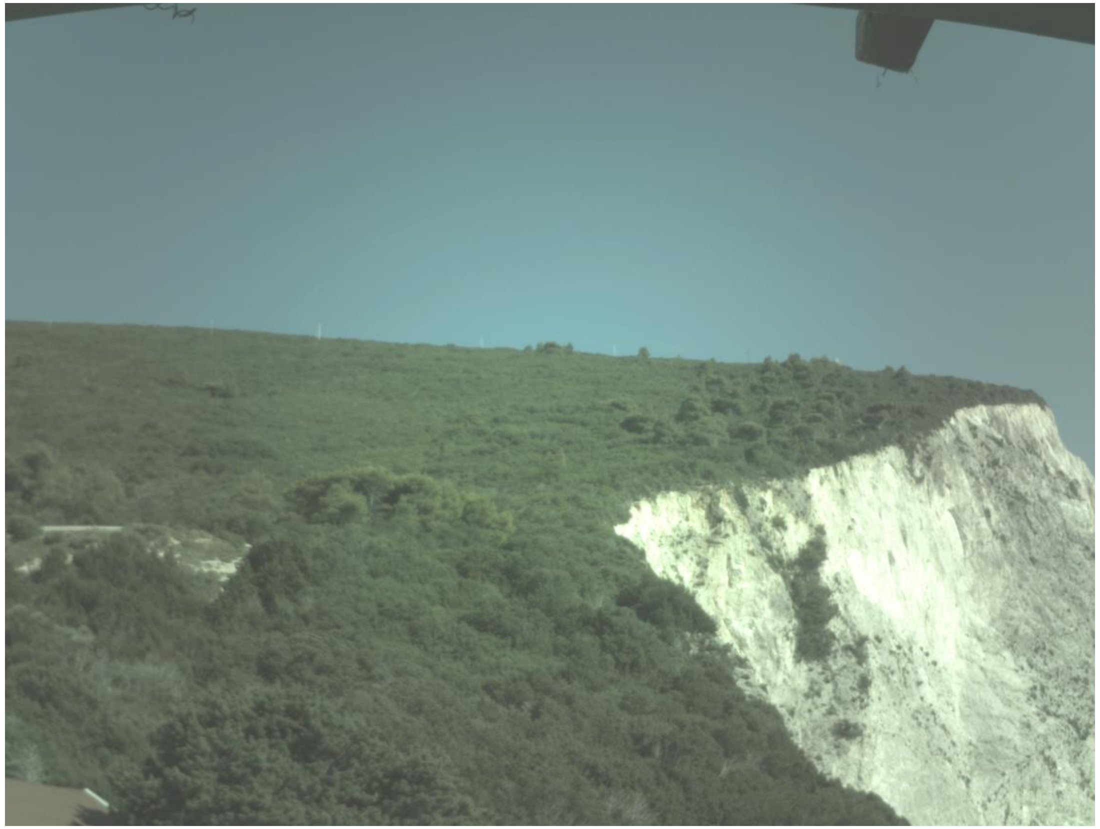


Before white balancing

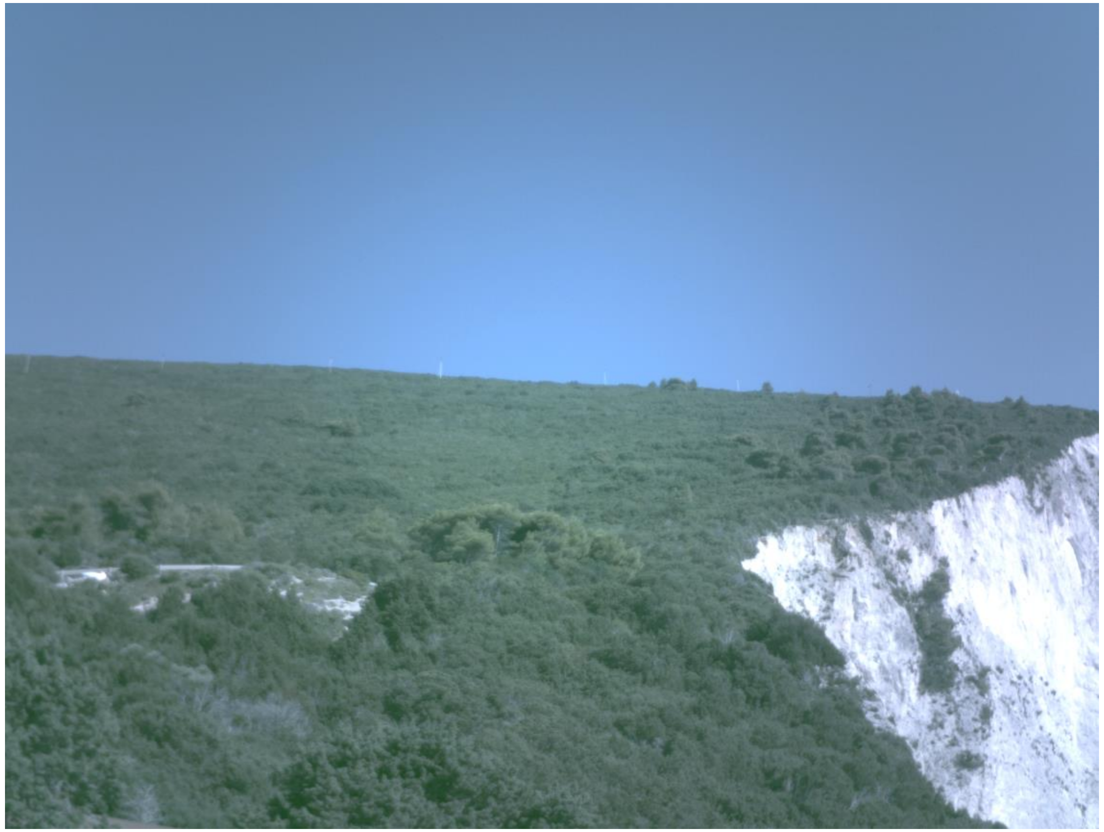


After white balancing

What the human eye sees

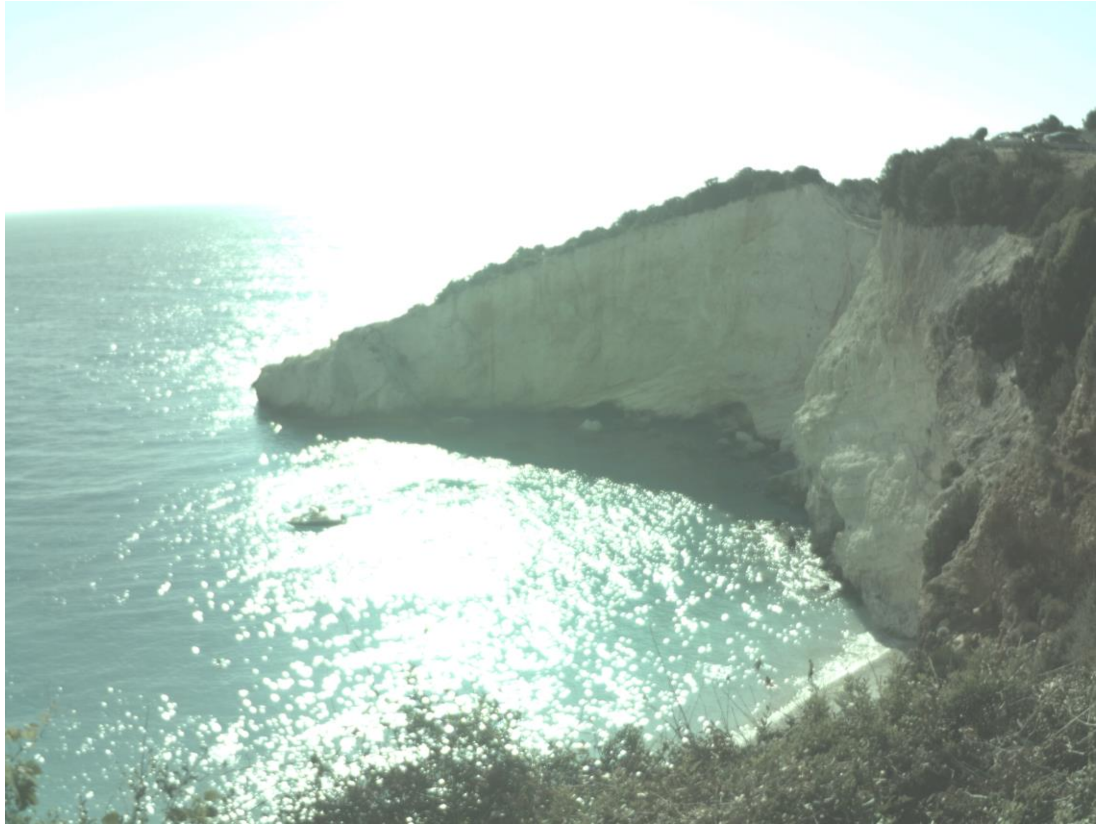


Before white balancing

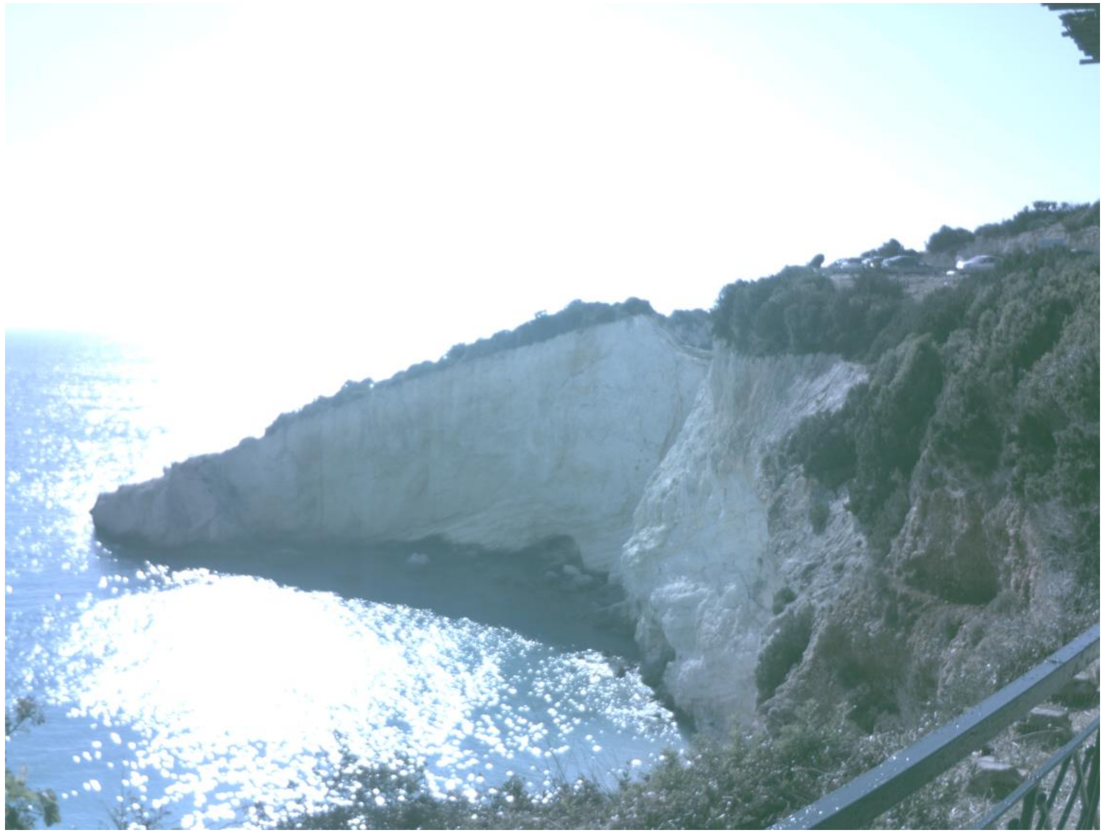


After white balancing

What the human eye sees

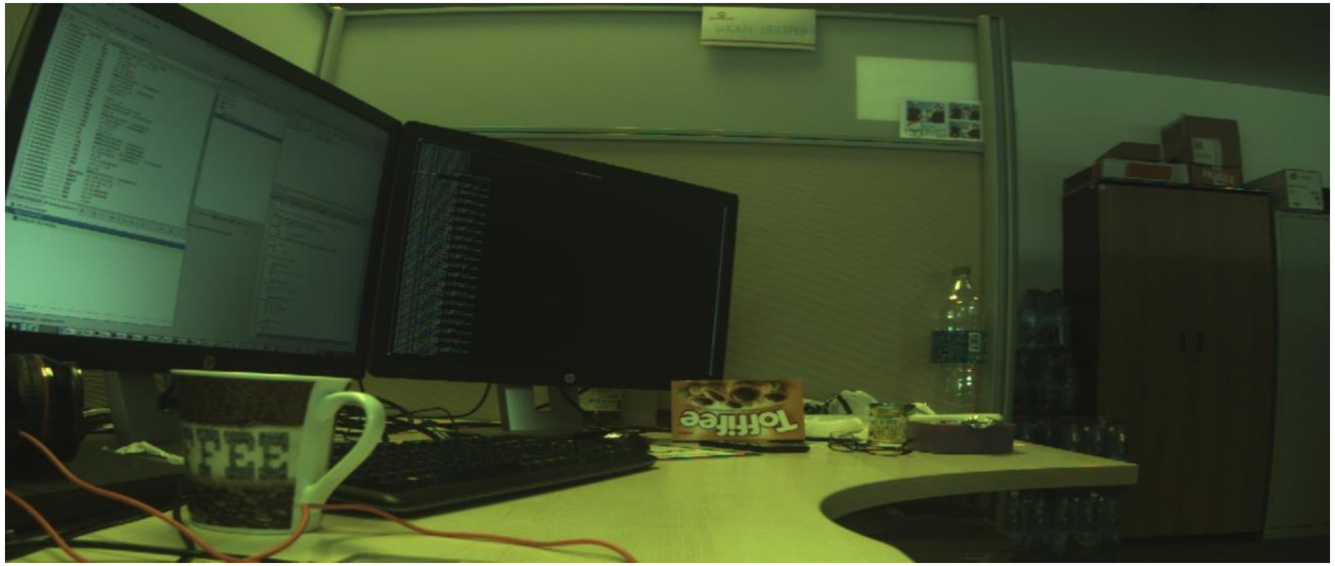


Before white balancing



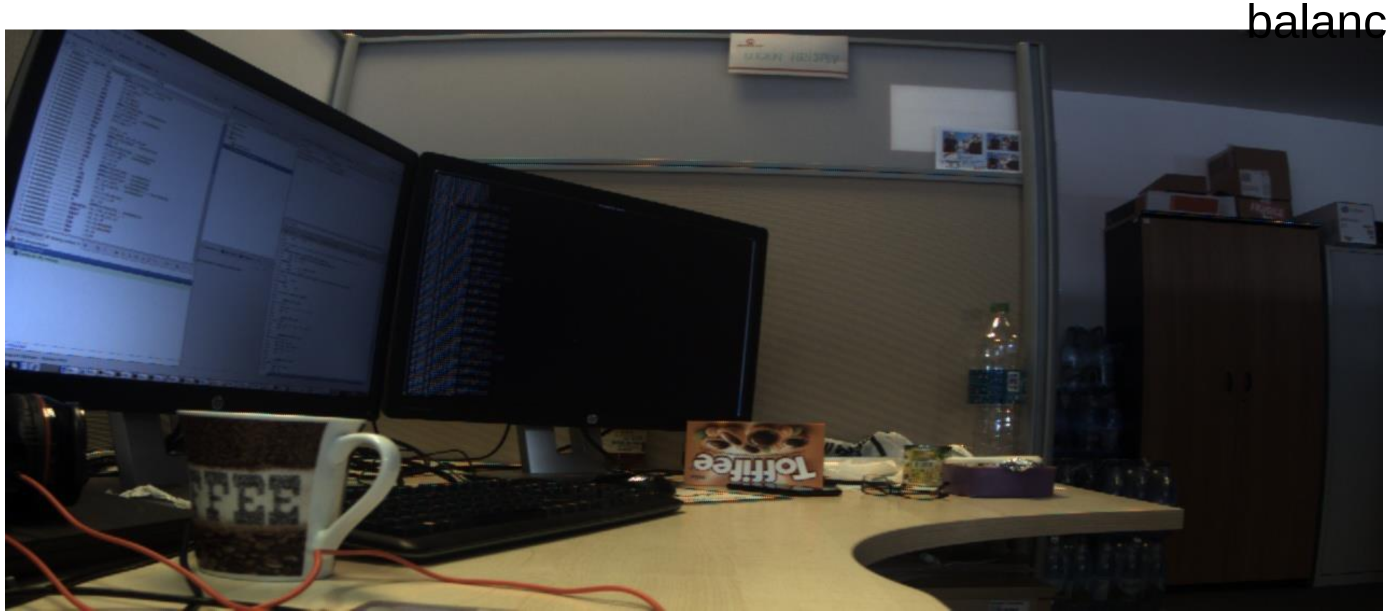
After white balancing

What the human eye sees



Before white balancing

After white balancing

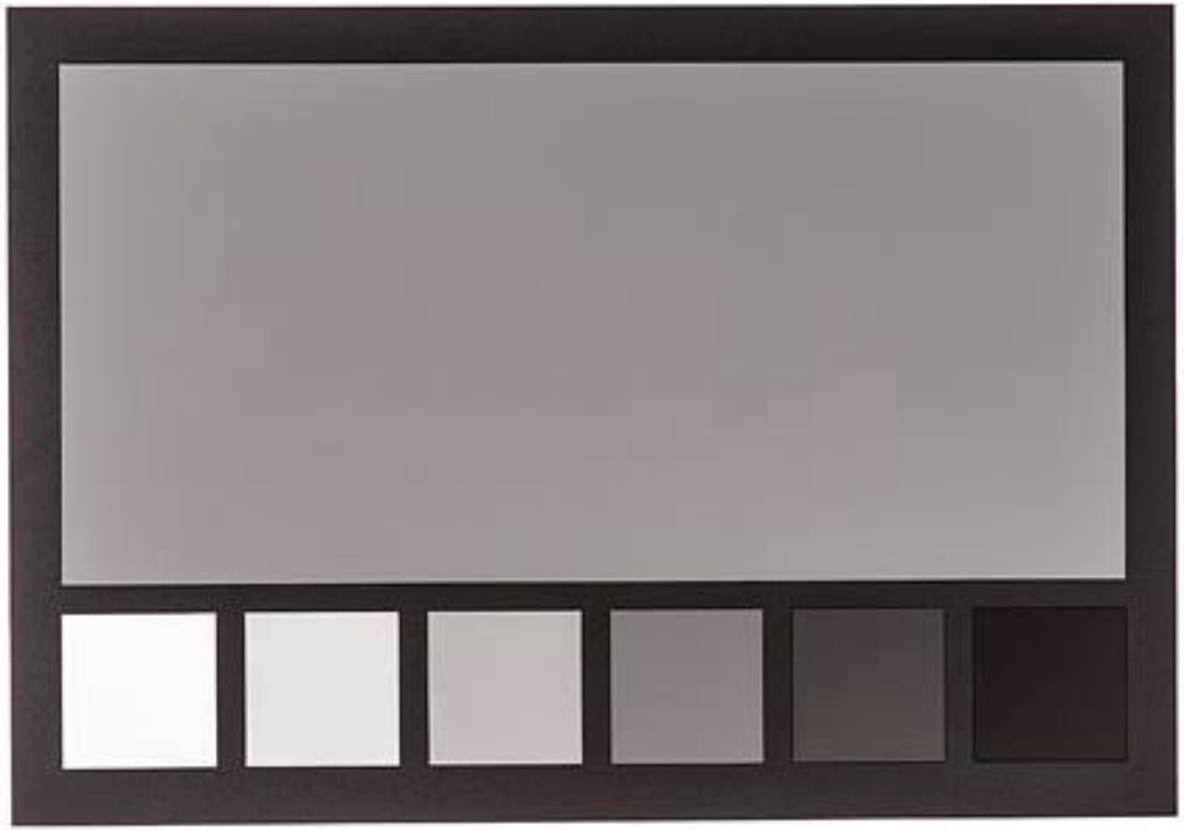


What can we do to adapt ?

We need to teach the sensor to adapt

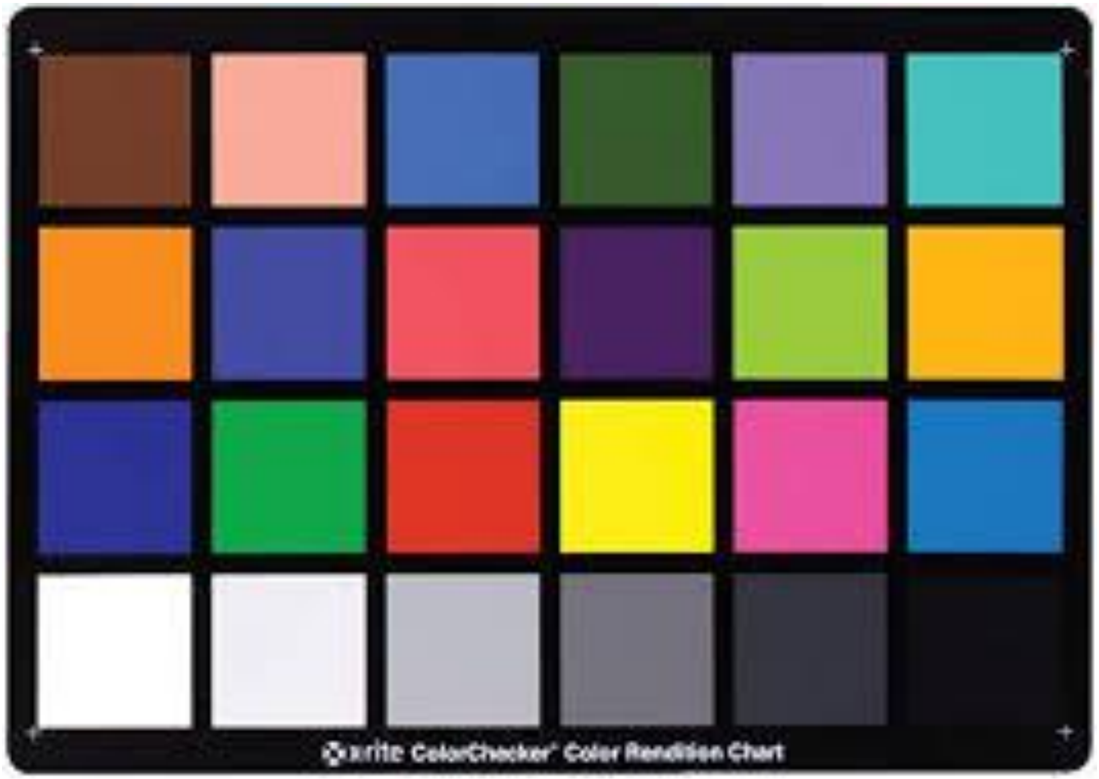
Question 1:

What is the **average** color of this photo ?



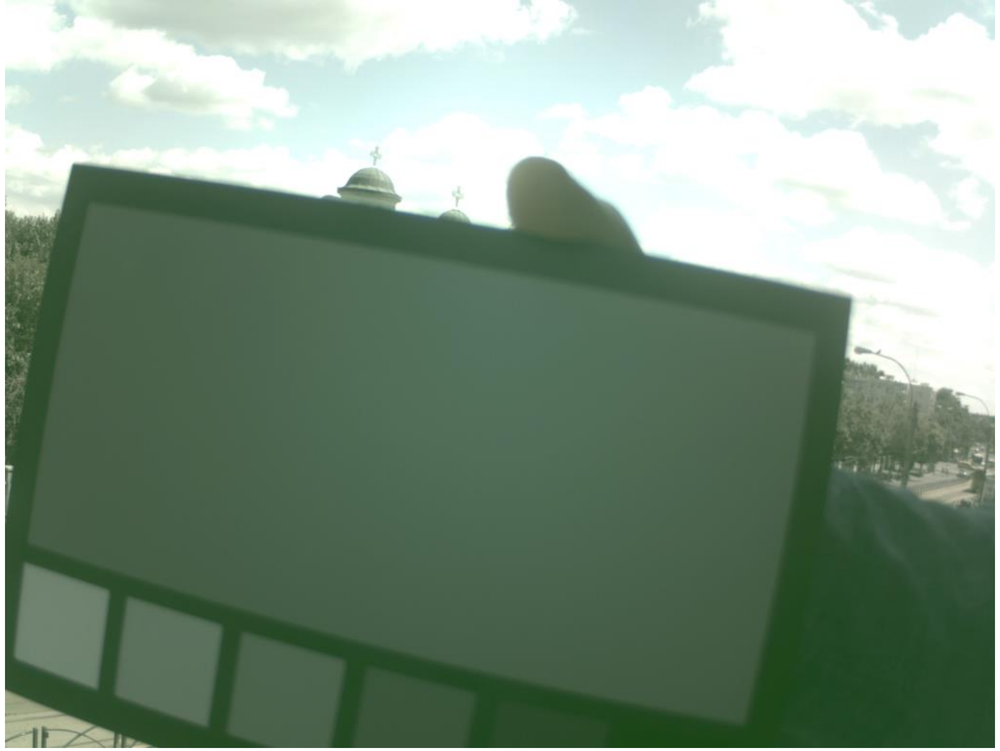
Question 2:

What is the **average** color of this photo ?



Grey world

We need to make sure that Grey is grey for us, and grey for the sensor. In the ambient light, of course...

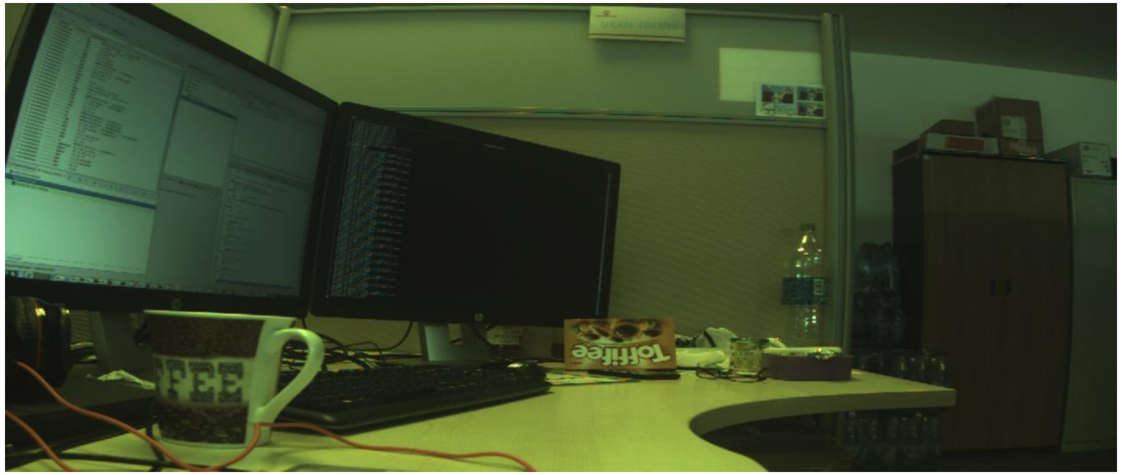


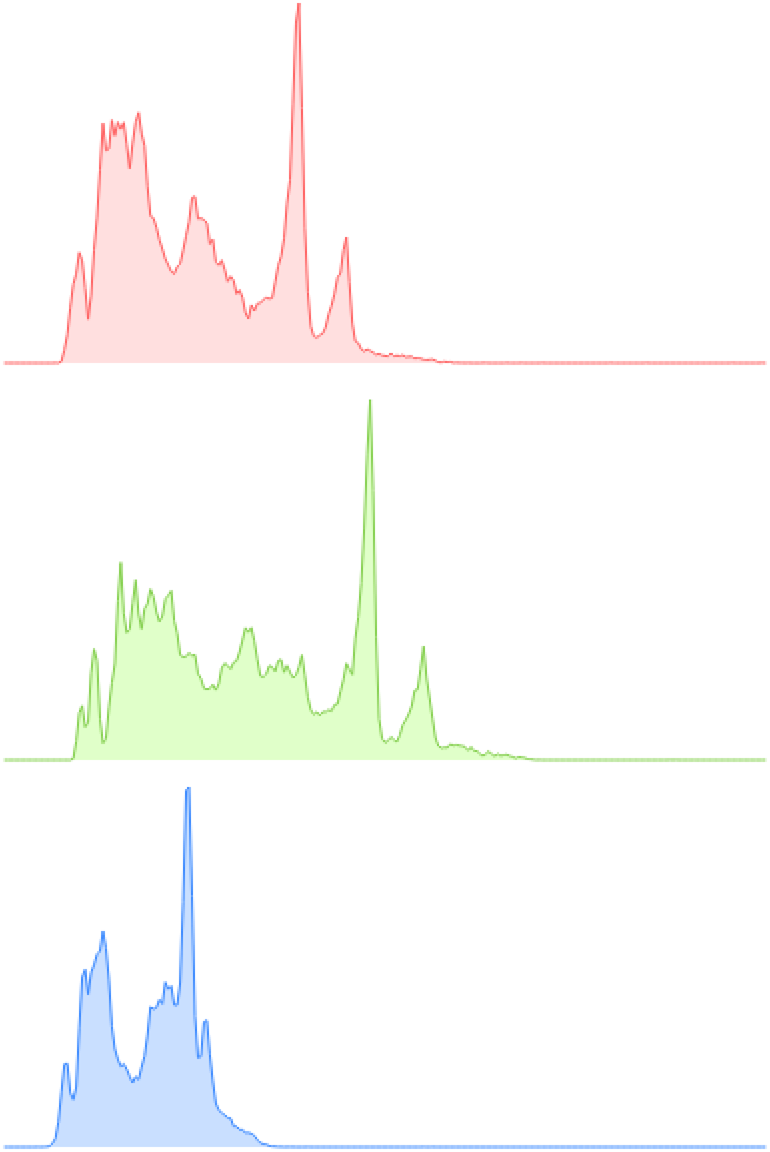
Algorithm inside the driver

We need four frames captured.

The hardware will compute a color histogram of each BAYER component

This will tell us how much of each color we have in the frame.
Will count the number of pixels versus the pixel value.





By histogram, this is greenish, with low blue

Algorithm inside the driver

We need to apply Grey World: everything is Grey !

Adjust gains and offsets such that the histogram will be nearly identic for each component.

Compute the average of R, B, RG, BG. This is $\bar{K}$

Keep Green as a constant and adjust gains for red and blue, by dividing with the average,

for each channel, $K'_{i,j}$

$$K'_{i,j} = \frac{K_{i,j}}{\bar{K}}$$

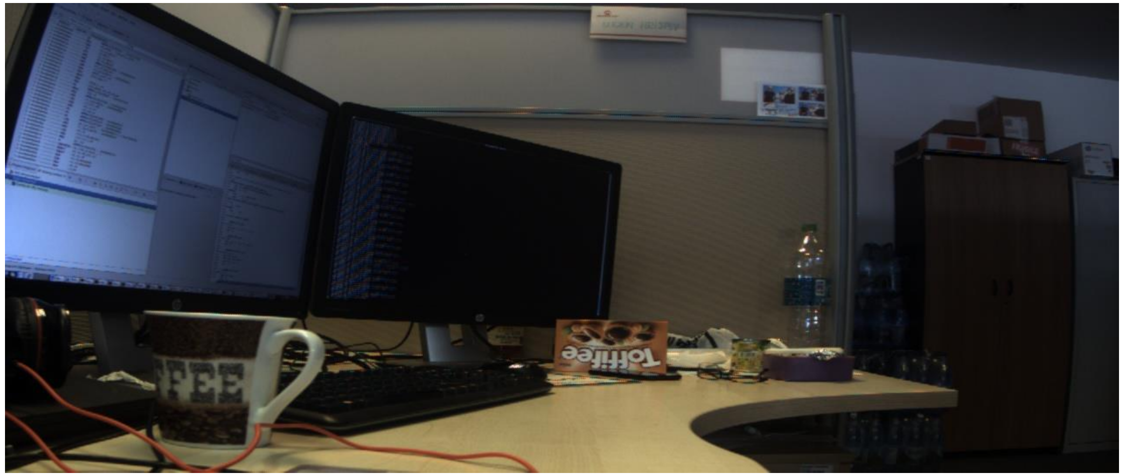
Ideally, we should see that the blue component will be aligned with the others.

Algorithm inside the driver

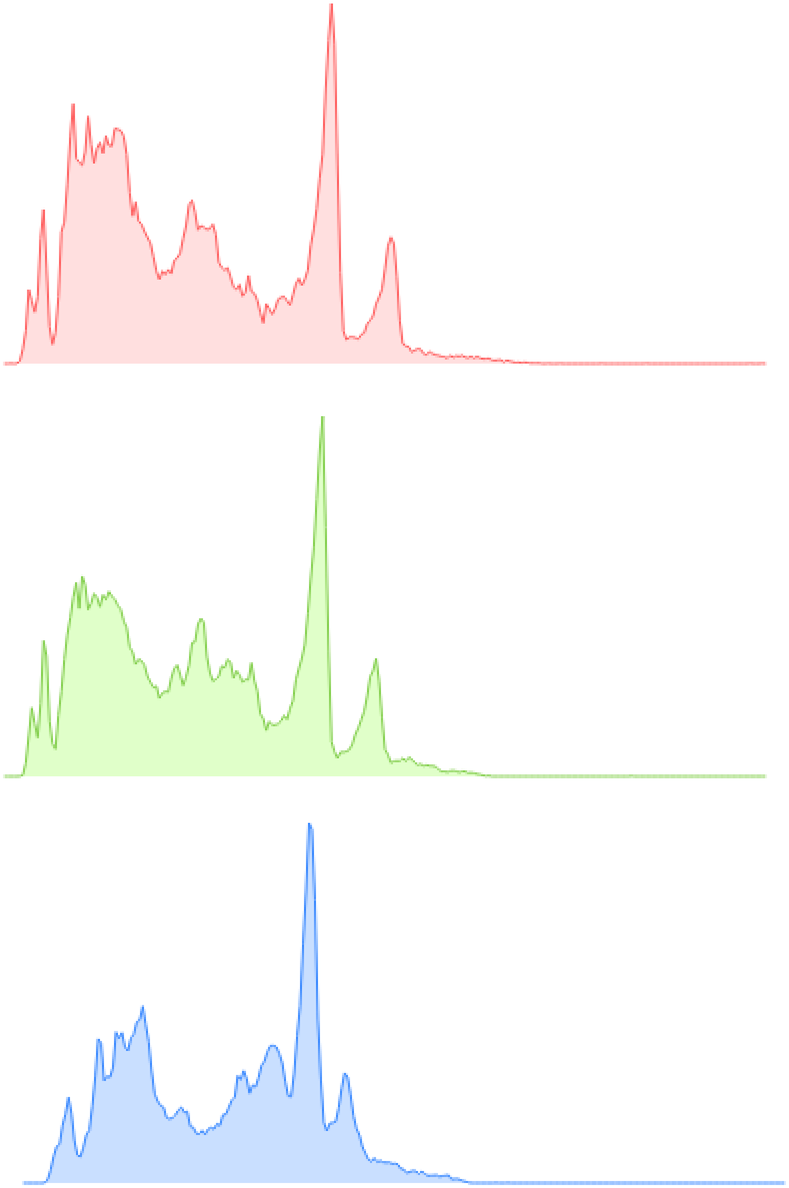
We need four frames captured.

The hardware will compute a color histogram of each BAYER component

This will tell us how much of each color we have in the frame.
Will count the number of pixels versus the pixel value.



Histogram is now aligned and stretched



Gains and offsets

V4L2 to the rescue:

```
User Controls

                 brightness 0x00980900 (int)    : min=-1024 max=1023 step=1 default=0 value=0 flags=slider
                   contrast 0x00980901 (int)    : min=-2048 max=2047 step=1 default=16 value=16 flags=slider
     white_balance_automatic 0x0098090c (bool)   : default=1 value=1 flags=update
            do_white_balance 0x0098090d (button) : flags=inactive, write-only, volatile, execute-on-write
                      gamma 0x00980910 (int)    : min=0 max=3 step=1 default=3 value=3 flags=slider
         red_component_gain 0x009819c0 (int)    : min=0 max=8191 step=1 default=512 value=512 flags=inactive, slider, volatile
        blue_component_gain 0x009819c1 (int)    : min=0 max=8191 step=1 default=512 value=512 flags=inactive, slider, volatile
   green_red_component_gain 0x009819c2 (int)    : min=0 max=8191 step=1 default=512 value=512 flags=inactive, slider, volatile
  green_blue_component_gain 0x009819c3 (int)    : min=0 max=8191 step=1 default=512 value=512 flags=inactive, slider, volatile
       red_component_offset 0x009819c4 (int)    : min=-4095 max=4095 step=1 default=0 value=0 flags=inactive, slider, volatile
      blue_component_offset 0x009819c5 (int)    : min=-4095 max=4095 step=1 default=0 value=0 flags=inactive, slider, volatile
 green_red_component_offset 0x009819c6 (int)    : min=-4095 max=4095 step=1 default=0 value=0 flags=inactive, slider, volatile
green_blue_component_offset 0x009819c7 (int)    : min=-4095 max=4095 step=1 default=0 value=0 flags=inactive, slider, volatile
```

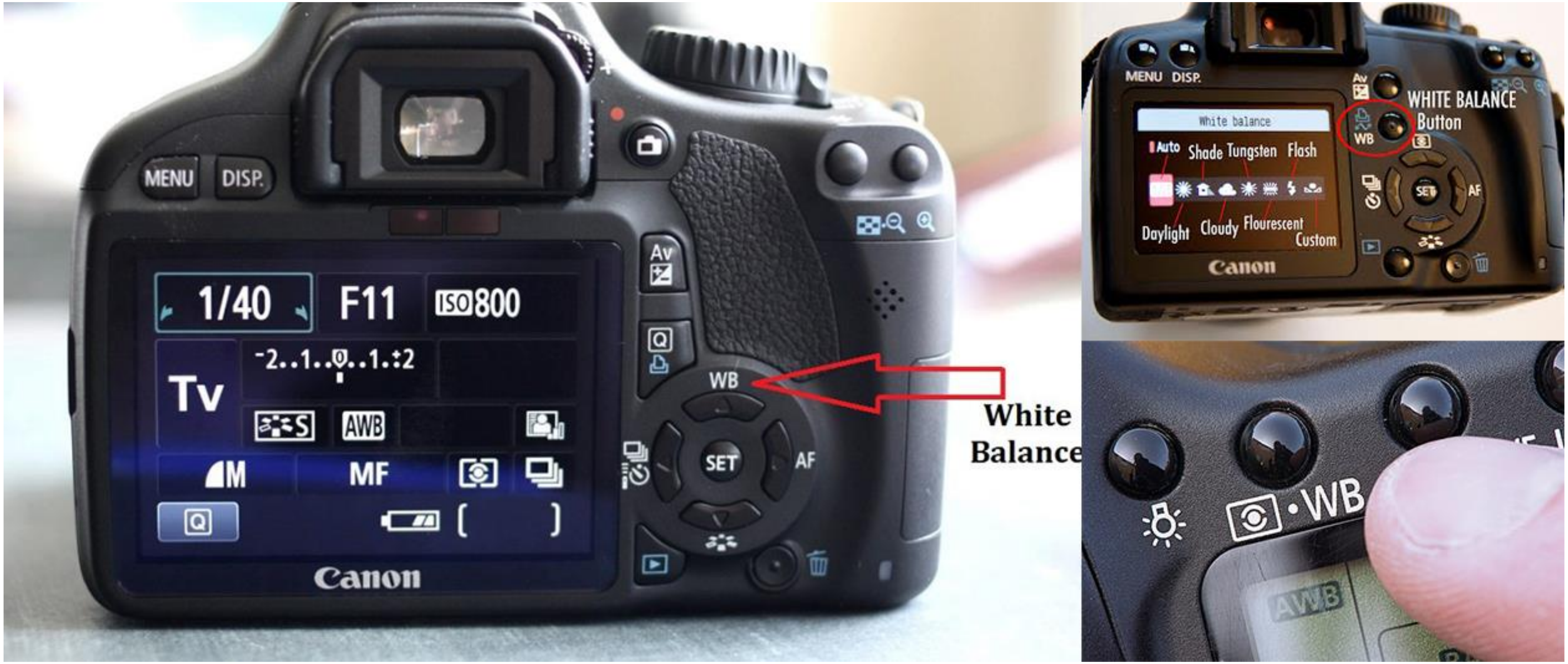
Do white balance button

```
v4l2-ctl –set-ctrl=do_white_balance=1
# [ 2528.410573] atmel-sama7g5-isc e1408000.xisc: Completed one time white-balance adjustment.
```

```
User Controls

                     brightness 0x00980900 (int)    : min=-1024 max=1023 step=1 default=0 value=0 flags=slider
                       contrast 0x00980901 (int)    : min=-2048 max=2047 step=1 default=16 value=16 flags=slider
        white_balance_automatic 0x0098090c (bool)   : default=1 value=0 flags=update
               do_white_balance 0x0098090d (button) : flags=write-only, execute-on-write
                          gamma 0x00980910 (int)    : min=0 max=3 step=1 default=3 value=3 flags=slider
             red_component_gain 0x009819c0 (int)    : min=0 max=8191 step=1 default=512 value=930 flags=slider
            blue_component_gain 0x009819c1 (int)    : min=0 max=8191 step=1 default=512 value=3198 flags=slider
       green_red_component_gain 0x009819c2 (int)    : min=0 max=8191 step=1 default=512 value=549 flags=slider
      green_blue_component_gain 0x009819c3 (int)    : min=0 max=8191 step=1 default=512 value=714 flags=slider
           red_component_offset 0x009819c4 (int)    : min=-4095 max=4095 step=1 default=0 value=-128 flags=slider
          blue_component_offset 0x009819c5 (int)    : min=-4095 max=4095 step=1 default=0 value=-48 flags=slider
     green_red_component_offset 0x009819c6 (int)    : min=-4095 max=4095 step=1 default=0 value=-152 flags=slider
    green_blue_component_offset 0x009819c7 (int)    : min=-4095 max=4095 step=1 default=0 value=-136 flags=slider
```

Do white balance button



Auto white balance (AWB)

Cameras will do this for you.

But how good is this ?

Is the scenery always grey ?

Experiment with your smartphone !

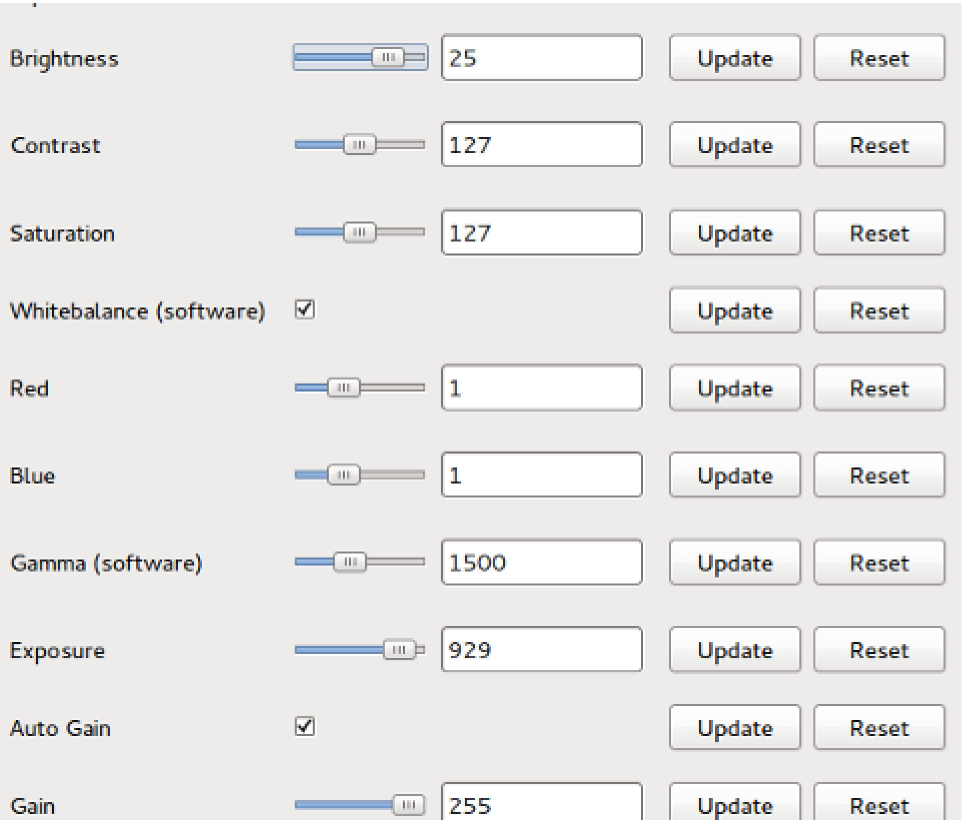
Other algorithms for white balance ?

Sensor correlation: find which of the elementary possible lights are there

Grey world variations: find the grey object in the scenery

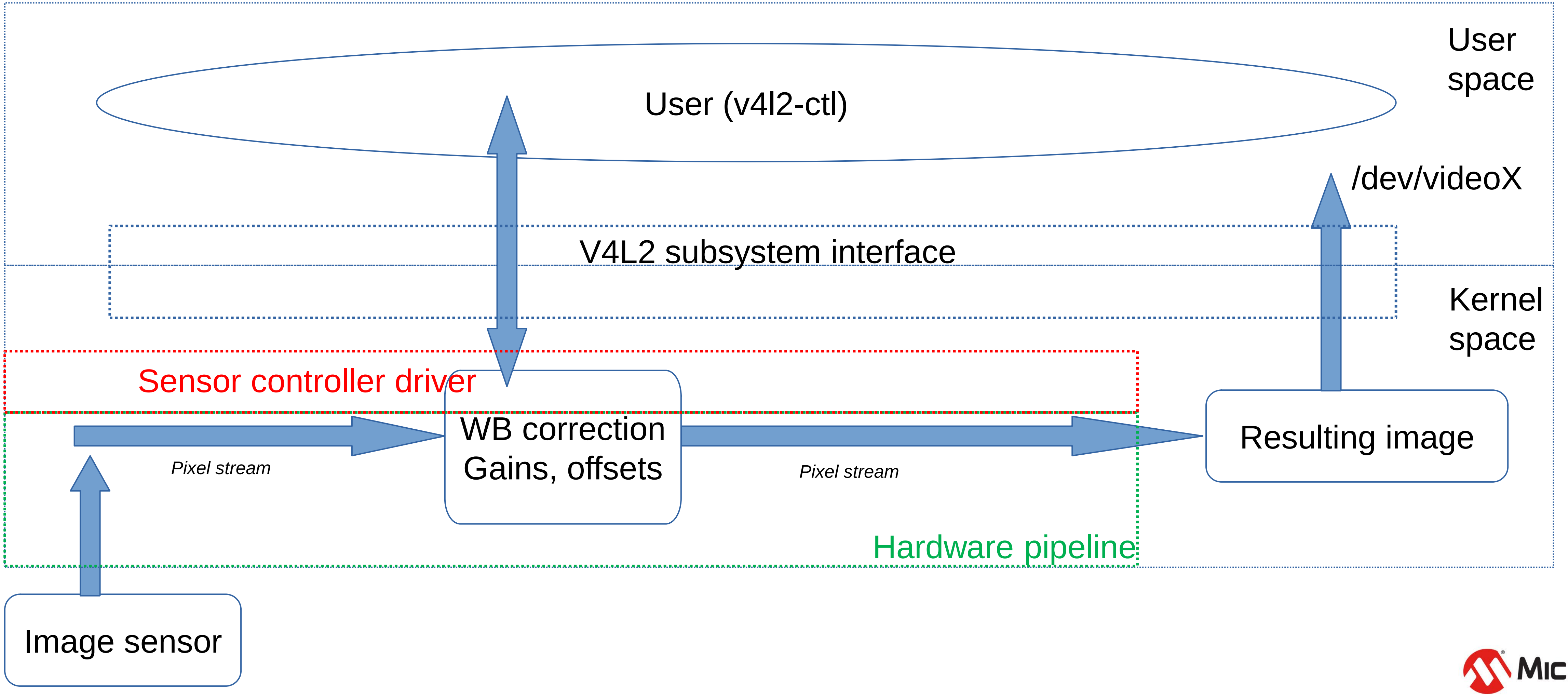
Photoshop WB: Black is black, White is white.

Manual tuning ! Towards Embedded Linux Camera...



Brightness	25
Contrast	127
Saturation	127
Whitebalance (software)	☑
Red	1
Blue	1
Gamma (software)	1500
Exposure	929
Auto Gain	☑
Gain	255

Inside the system



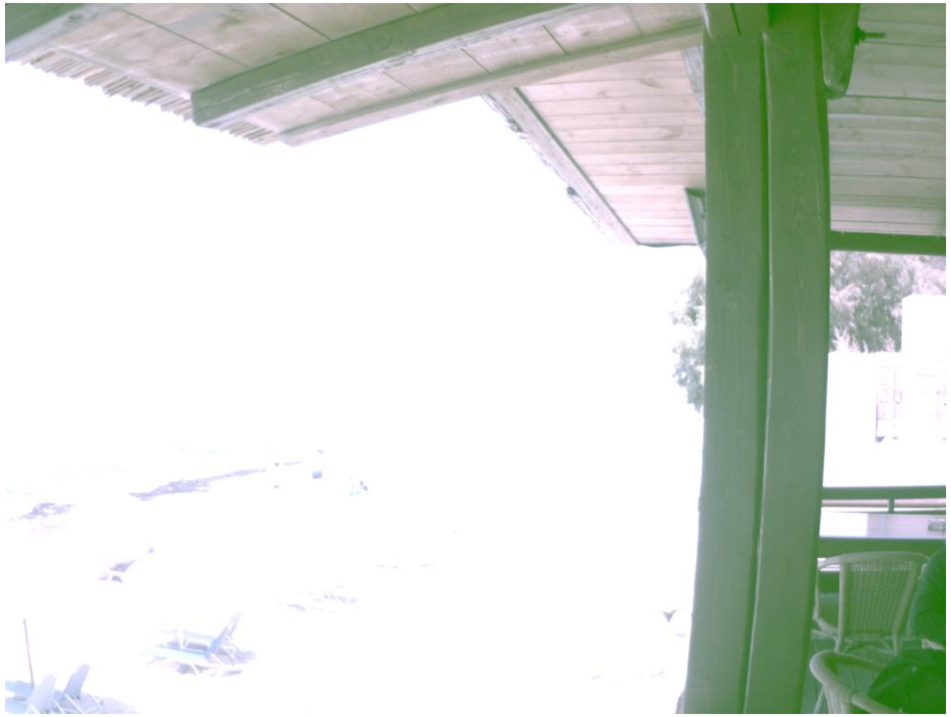
Light quantity

Does it matter how much light we capture ?

Exposure

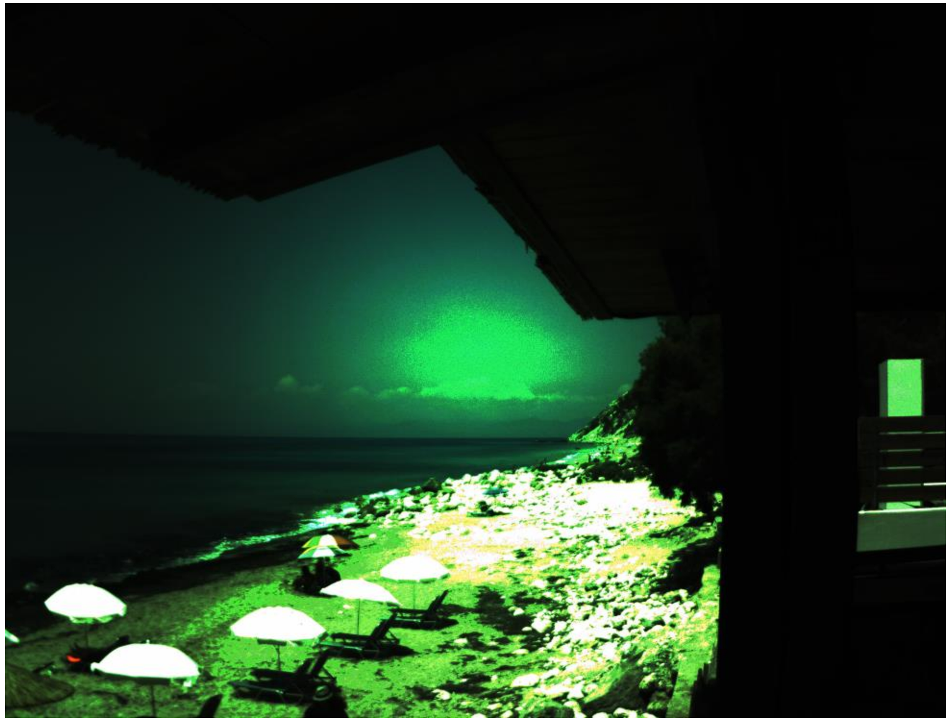
How much light do we need to capture ?

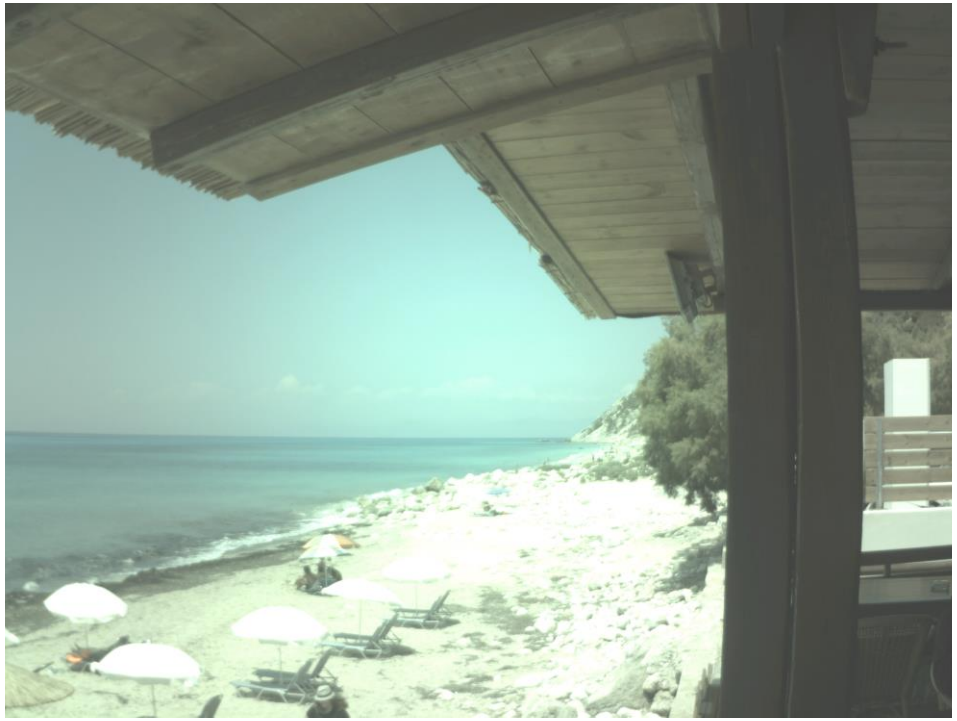
Too much light



Again, trade-off

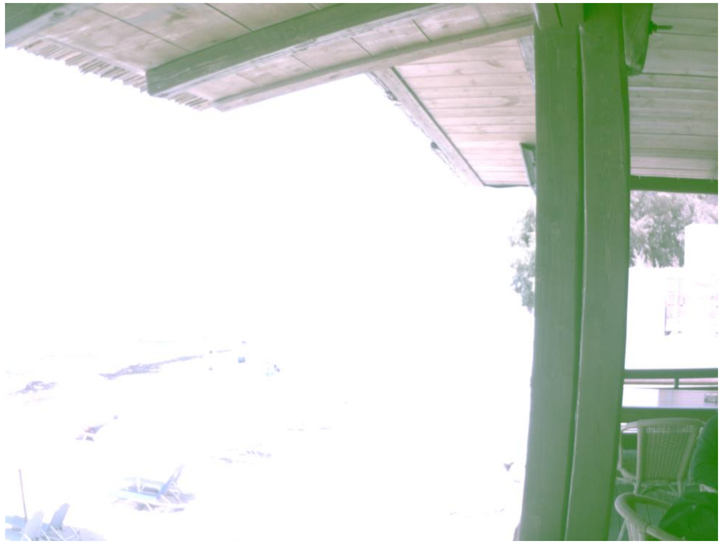
Too little light

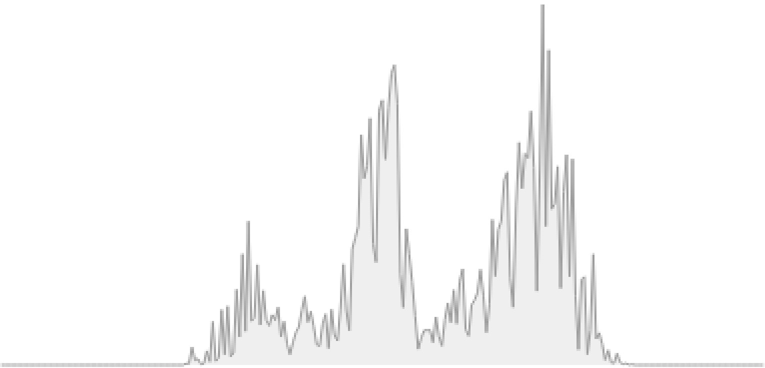


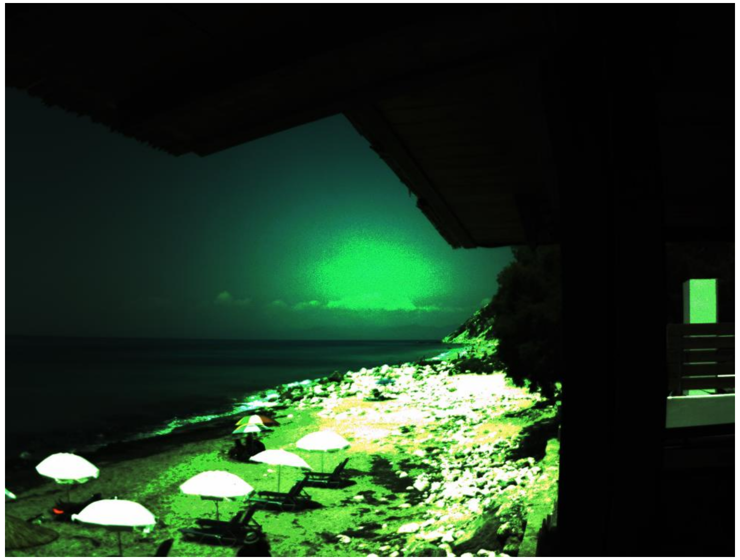


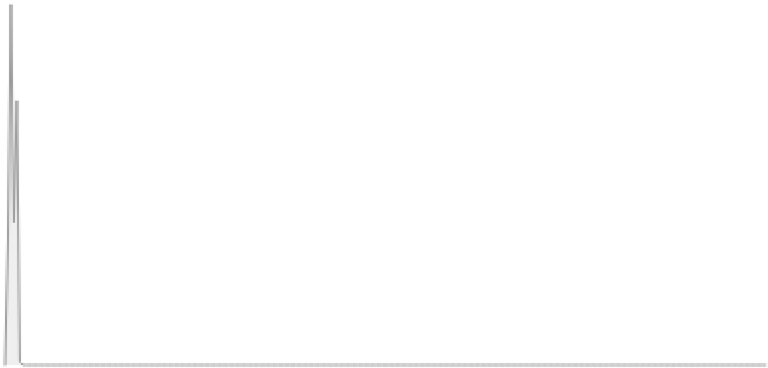
Exposure

Again, histogram comes to our aid

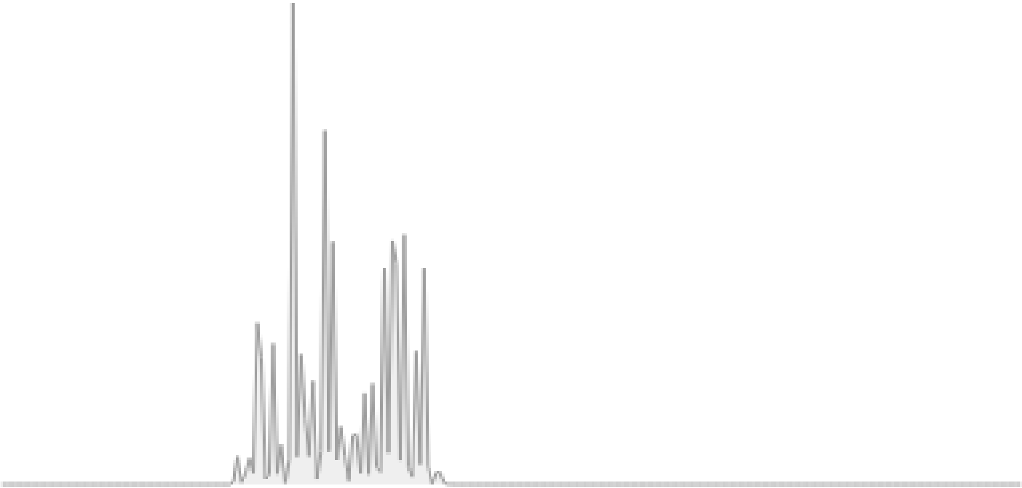








Exposure



Autoexposure: adapt exposure to fit the histogram

Exposure

```
# v4l2-ctl -L -d /dev/v4l-subdev1

User Controls

                   exposure 0x00980911 (int)    : min=14 max=8333 step=1 default=8333 value=8260
                       gain 0x00980913 (int)    : min=256 max=46088 step=1 default=5120 value=5120
              vertical_flip 0x00980915 (bool)   : default=0 value=0
```

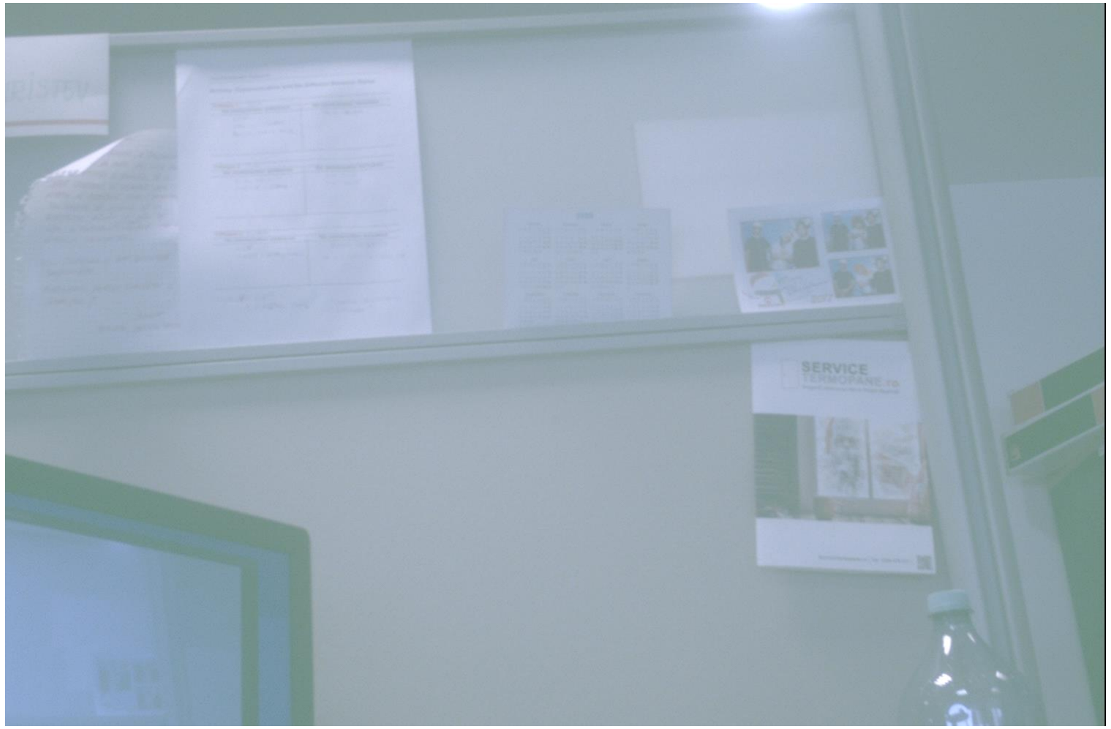
V4L2 controls directly to the subdevice

Brightness

```
# v4l2-ctl -L

User Controls
                   brightness 0x00980900 (int)    : min=-1024 max=1023 step=1 default=0 value=256
flags=slider
```

Positive brightness applied

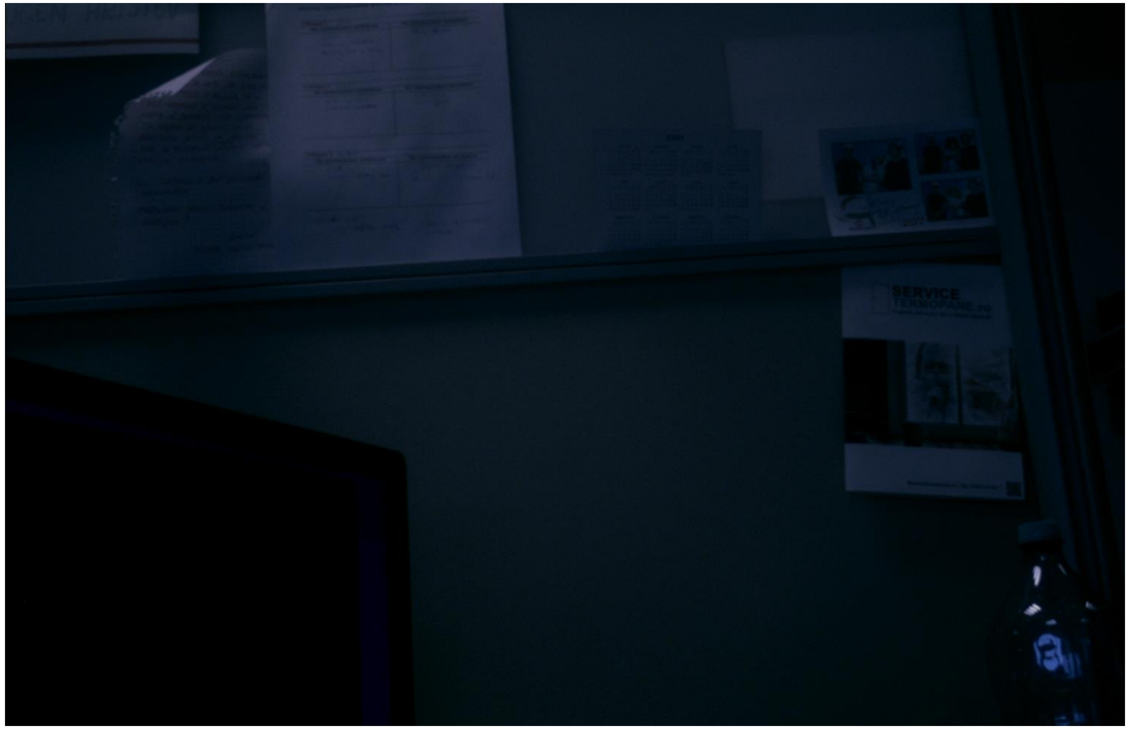


Brightness

```
# v4l2-ctl -L

User Controls
                     brightness 0x00980900 (int)    : min=-1024 max=1023 step=1 default=0 value=-256
flags=slider
```

Negative brightness applied

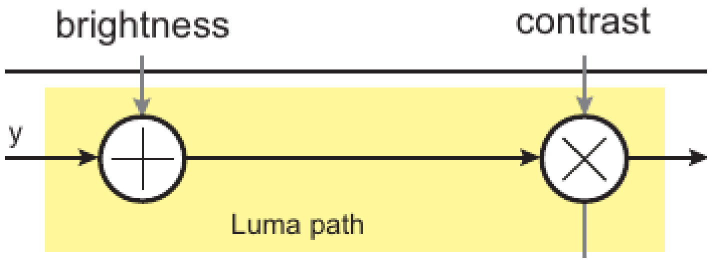


Brightness vs exposure

Why do we need brightness if we have exposure ?

Is the other way around true ?

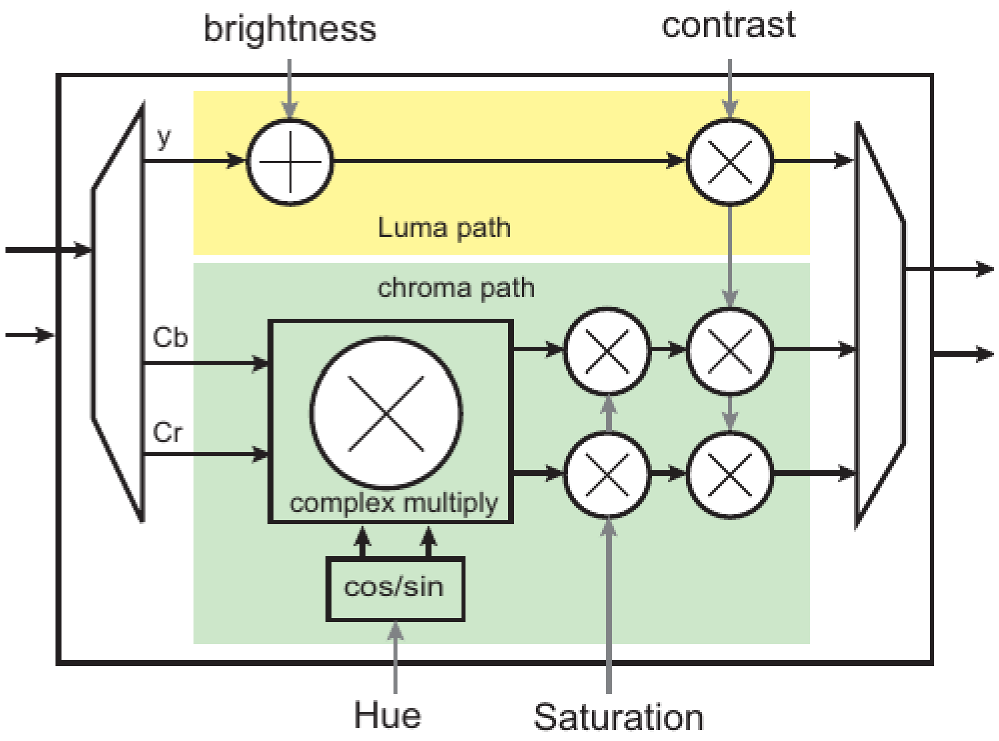
What happens with pixel data ?



How can we adjust colors

Can we adjust the ratio between the chroma channels ?

To make it easier, use the YUV representation : adjust blue Chroma and red Chroma



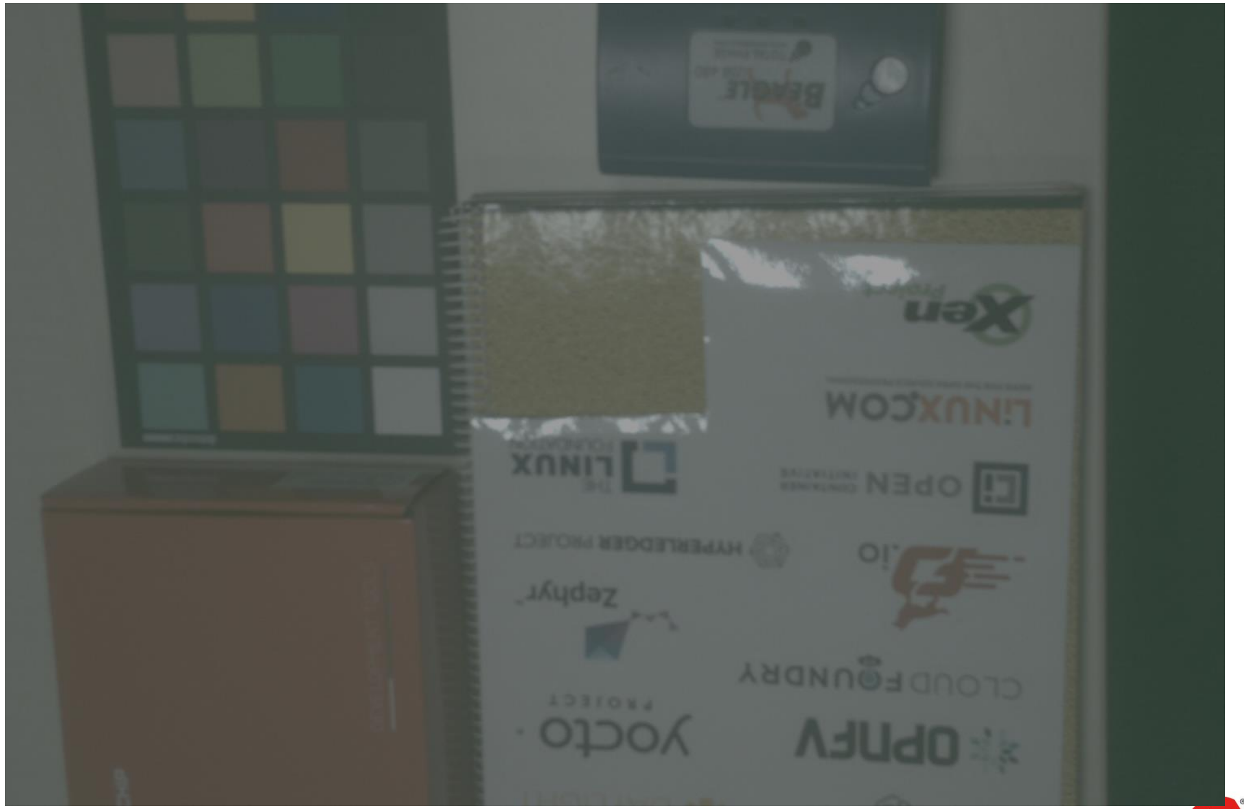
V4L2 controls can alter this hardware block

Contrast

```
# v4l2-ctl -L

User Controls
                     contrast 0x00980901 (int)    : min=-2048 max=2047 step=1 default=16 value=8
flags=slider
```

Low contrast applied

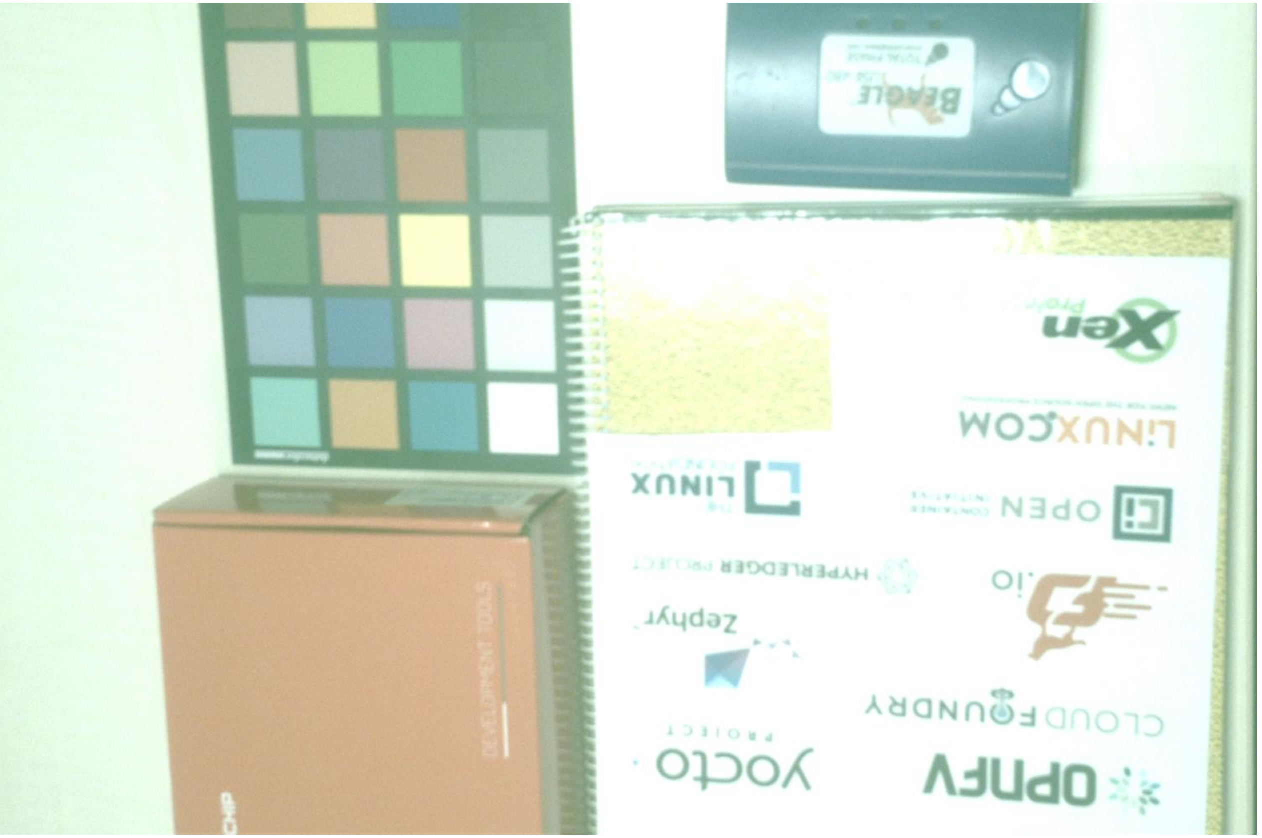


Contrast

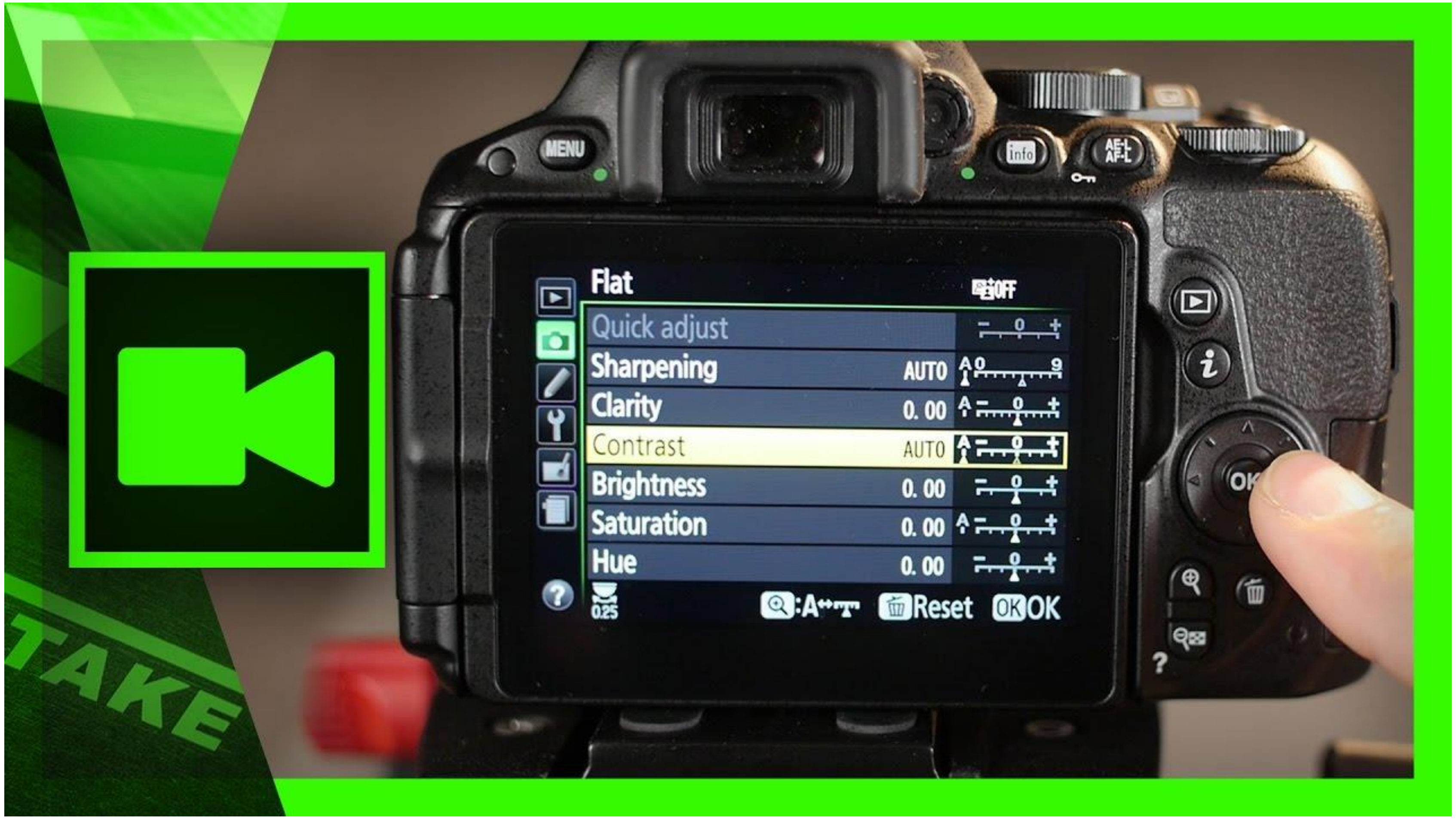
```
# v4l2-ctl -L

User Controls
                   contrast 0x00980901 (int)    : min=-2048 max=2047 step=1 default=16 value=8
flags=slider
```

High contrast applied



Brightness and Contrast



Flat

Quick adjust

Sharpening AUTO

Clarity 0.00

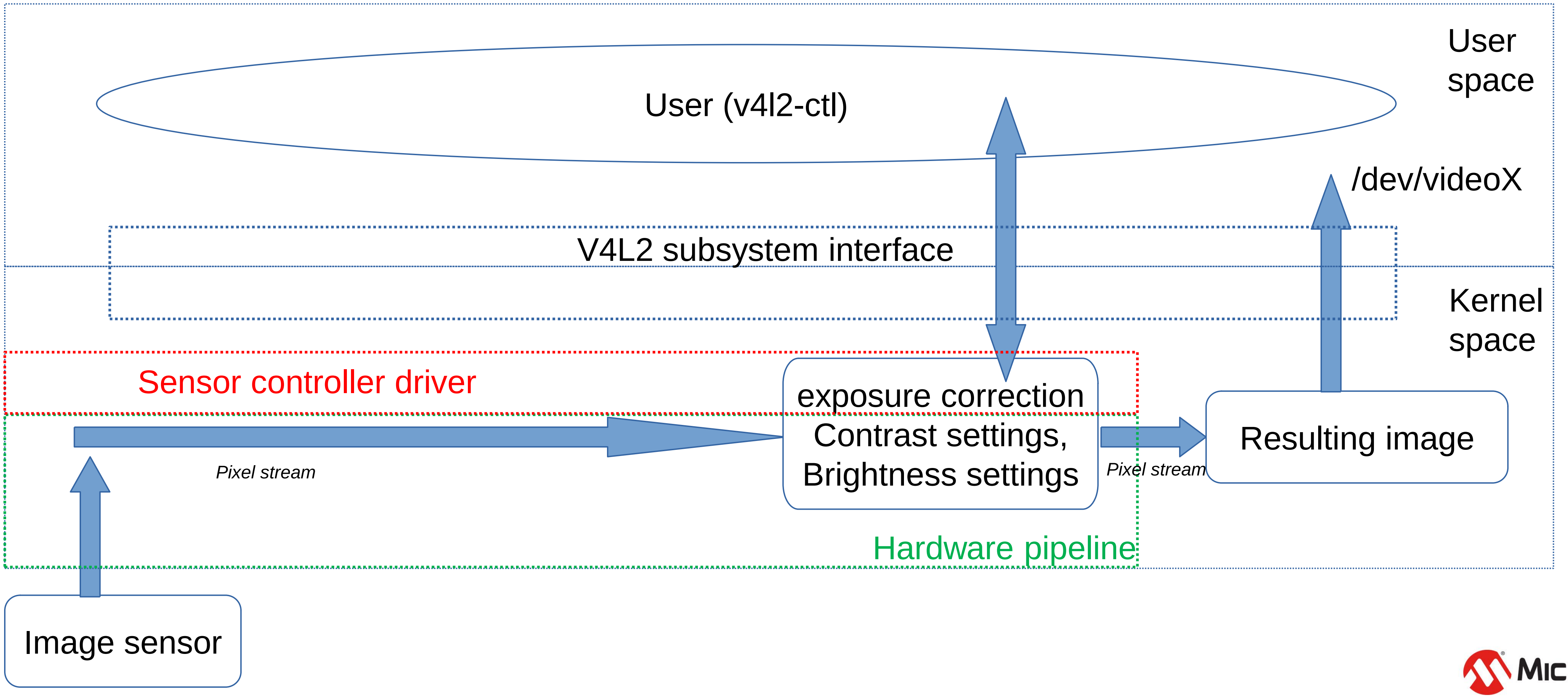
Contrast AUTO

Brightness 0.00

Saturation 0.00

Hue 0.00

Inside the system



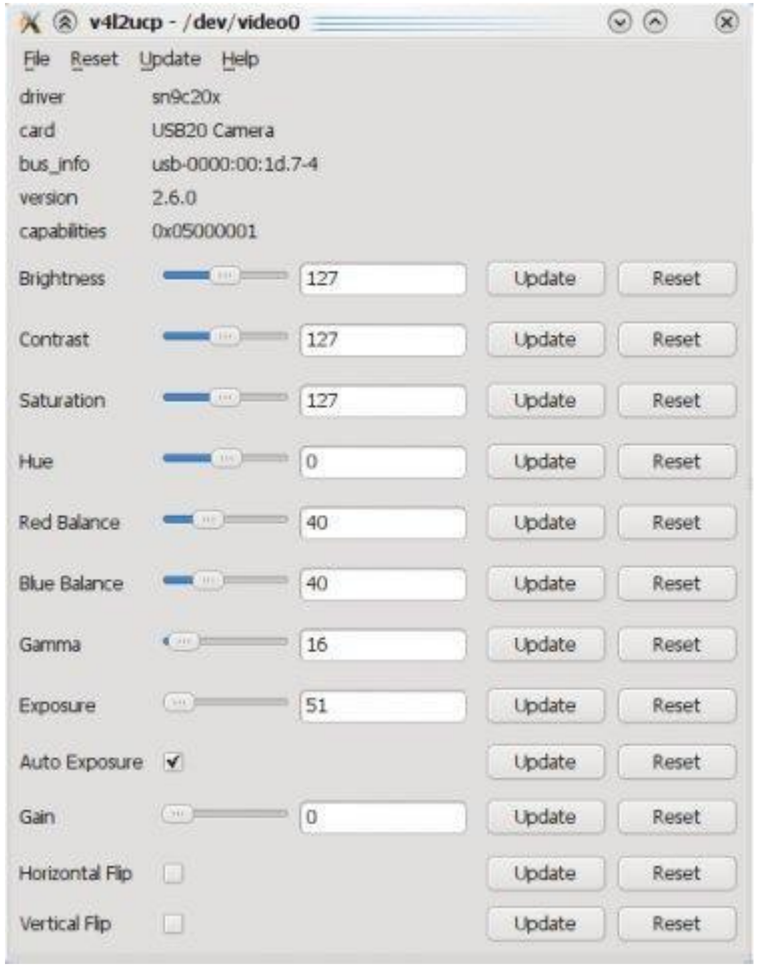
Summary

Digital sensors need tuning

We can use a dedicated pipeline to achieve this

We can control the pipeline using Linux and V4L2

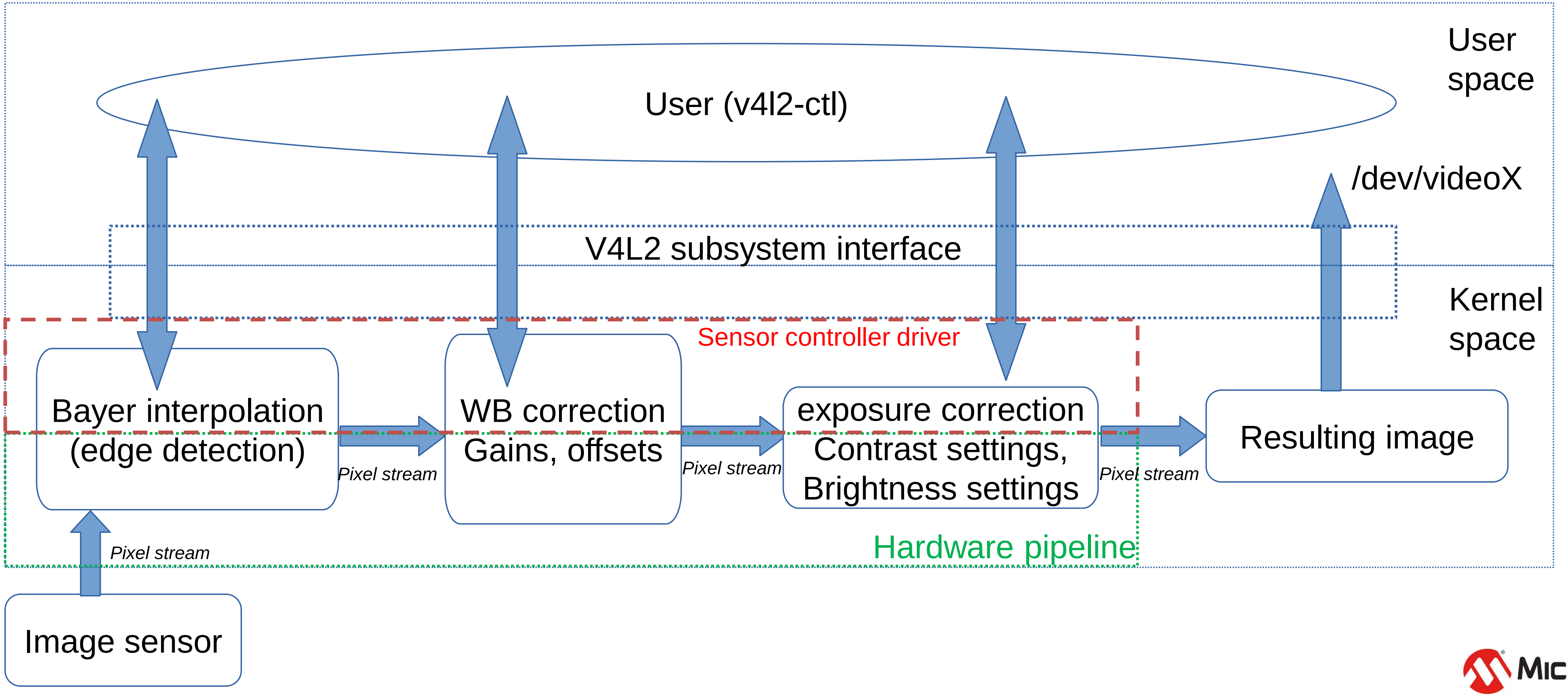
We can have an embedded Linux Camera ,
with sliders, buttons as interface



driver sn9c20x
card USB20 Camera
bus_info usb-0000:00:1d.7-4
version 2.6.0
capabilities 0x05000001

Brightness 127
Contrast 127
Saturation 127
Hue 0
Red Balance 40
Blue Balance 40
Gamma 16
Exposure 51
Auto Exposure ☑
Gain 0
Horizontal Flip □
Vertical Flip □

Summary


User space
User (v4l2-ctl)
/dev/videoX
V4L2 subsystem interface
Kernel space
Sensor controller driver
Bayer interpolation (edge detection)
WB correction Gains, offsets
exposure correction Contrast settings, Brightness settings
Resulting image
Pixel stream
Pixel stream
Pixel stream
Pixel stream
Hardware pipeline
Image sensor

Demo time

Questions?

Resources and Availability

- Atmel ISC driver which served as a base for this presentation

https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/drivers/media/platform/atmel

- Images from this presentation are licensed under CC-BA-SY license

- Where license was not specified, some images source link was provided
- Otherwise, all images are personal property of the author, and were taken with SAMA7G5-EK board with SAMA7G5 SoC,

using image sensor Sony imx219 and Sony imx274

https://arindamdhar.com/wp-content/uploads/2017/07/wb-button.jpg
https://www.pathpartnertech.de/wp-content/uploads/2020/03/Figure-1.jpg
https://starecat.com/content/wp-content/uploads/light-temperature-comparison-visual-1000k-to-10000k.jpg
https://www.xrite.com/categories/calibration-profiling/colorchecker-classic
https://a.fsdn.com/con/app/proj/v4l2ucp/screenshots/228307.jpg/max/max/1
https://i1.wp.com/digital-photography-school.com/wp-content/uploads/2016/01/
PlusMinusButton.jpg?fit=750%2C430&ssl=1
https://www.researchgate.net/publication/235350557_Combining_Gray-World_assumption_White-Point_
correction_and_power_transformation_for_automatic_white_balance
https://i.ytimg.com/vi/ZK0KX4uKhLM/maxresdefault.jpg