Software ISP FOSS Support for MIPI Cameras

Hans de Goede, Red Hat
Bryan O'Donoghue, Linaro

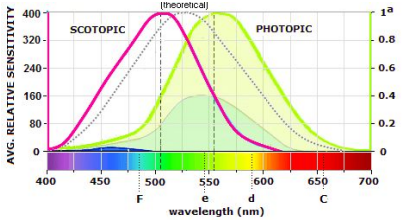
Introductions

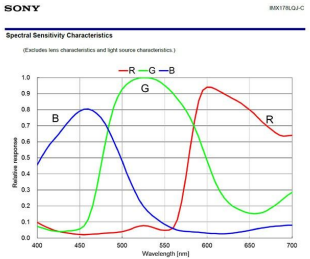
- Hans
  - Principal sw engineer @ Red Hat (Laptop HW enablement)
  - Kernel subsystem maintainer for drivers/platform/x86
  - https://git.kernel.org/pub/scm/linux/kernel/git/hansg/linux.git/
  - hansg @ #linux-media #libcamera #fedora-workstation
- Bryan
  - Kernel engineer @ Linaro (Qualcomm Landing Team)
  - Kernel media-committers maintainer - Qualcomm CAMSS, Iris, Venus
    - https://git.codelinaro.org/bryan.odonoghue/kernel
    - https://gitlab.freedesktop.org/linux-media/users/bodonoghue.git
    - https://github.com/0xB0D
  - bryanodonoghue @ #linux-media #linux #aarch64-laptops #libcamera #linux-msm

Topics

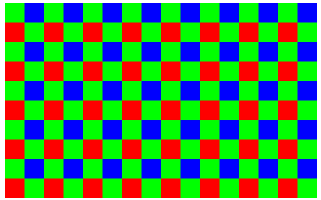
- Bayer Encoding
- CSI2 MIPI Cameras
- What is a Hard ISP - what are the 3As ?
- SoftISP in Libcamera - what’s the problem we are trying to solve ?
- Pipewire and Libcamera
- GPUISP
- Demos

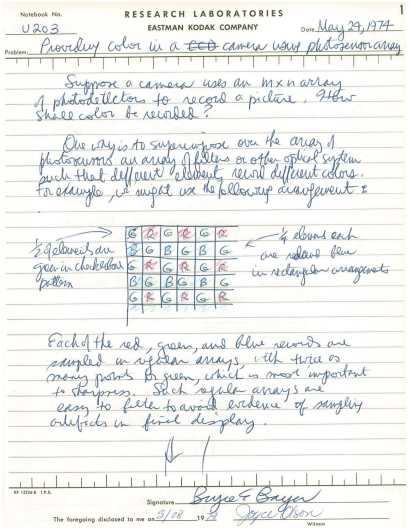
Bayer Encoding





- Human eye is composed of “rods” and “cones”
  - Rods Scotopic
  - Cones Photopic - sensitive to light around 550 nm
  - Most sensitive to “greenish yellow” colours
- Most RGB sensors don’t capture RGB
  - Contain monochrome sensors
- Sensors overlaid with a “mosaic” pattern of Red, Green, Blue
  - Bruce Bayer Eastman Kodak 1974
  - Bayer encoding
- Problem - each pixel contains only one colour
  - Approximate based on proximate pixels
  - Called “interpolation”
- Methods
  - Label
  - Nearest
  - Bilinear
  - Malar-He-Cutler





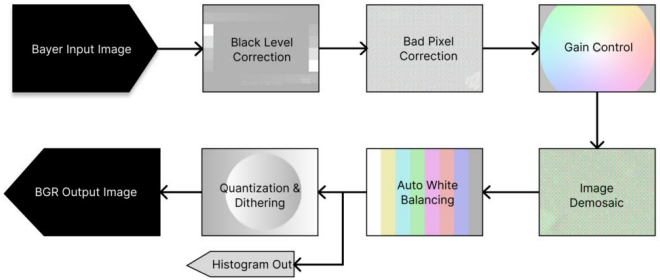
THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE

CSI2 MIPI Cameras

- USB - UVC Cameras
  - What you might consider a “standard” camera
  - Enormously expensive and power consuming
  - USB is an entire data communications protocol
- CSI2 MIPI Sensors
  - Proliferation of these sensors due to mobile phone economies of scale
- CSI2 standard
  - A PHY level analogue signal
- Unprocessed output images from most MIPI sensors
  - Various type of Bayer encoded data
  - Libcamera
    - Integration of MIPI Cameras into standard Linux user-space
- Raspberry Pi
- Intel IPU3 / IPU6 / IPU7
- Qualcomm Linux on upstream stacks

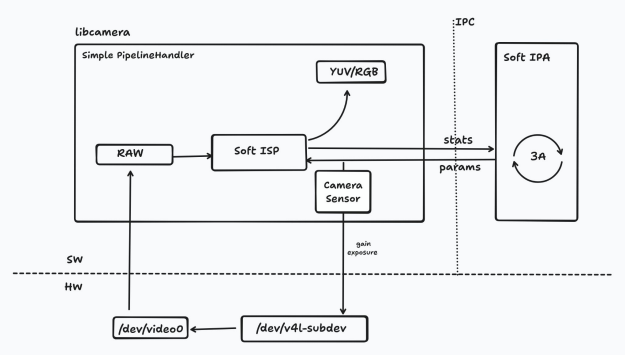
What is a HardISP?

- HardISP
  - Hardware Image Signal Processor
  - A specialised silicon or firmware block - usually both
  - Principle of data locality - process close to sensor input
- Debayers
- Implements 3As
  - Autofocus (AF)
  - Auto White Balance (AWB)
  - Auto Exposure / Gain (AG)
    - Usually MIPI sensors have an analogue and digital gain
- More advanced “secret sauce” algorithms
  - Sensor tunings (joking but not joking)
  - Skin tones
  - Low light noise reductions
  - Contrast
  - Lens flare reductions



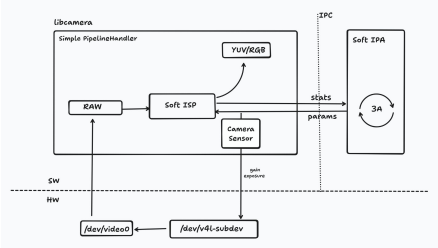
CPUISP

- Libcamera an open source library to handle v4l2 complex cameras
- Different “pipelines” handle different hardware implementations
  - RPI has a full hardware ISP pipeline
  - A generic simple pipeline handler for RAW bayer cameras exists
- A software ISP implemented in the CPU has been upstreamed
  - Debayer histogram and parameters generated and applied in CPU
  - Auto White Balance
  - Gain
  - Gamma correction
- Performance and features
  - Works on all tested Qualcomm platforms
    - sdm845, sm8250, sc8280xp, x1e80100
  - Works on x86 IPU6
  - Too much for lower-end ARM systems
    - TI Sitara AM64x
    - i.MX8M



GPUISP

- GPUs are massively parallel pixel pixel processing engines
- Libcamera already contains simple debayer shaders in a QT application
- Shaders debayer 8 bit RAW as well as 10/12 bit MIPI
  - Vertex/Fragment shaders already upstream and somewhat validated
  - Objective to reuse these shaders unmodified
- GLES 2.0 not OpenGL
  - To support the greatest range of hardware possible
  - Places constraints on some of the techniques we can use to attack the problem
  - eGL 1.0 was our first target but for a number of reasons it is undesirable
  - GLES 2.0 released March 2007 => nearly 20 years of supported hardware
- Implement features CPU can’t handle
  - Denoise
  - Dead pixel correction
  - Lens Shading



GPUISP - results

- Qualcomm sm8250 RB5 - IMX577 @ 4048 x 3040
  - CPUISP + CCM:
    - 2 FPS CPU usage > 100% single core pulls about 9 watts
  - GPUISP v1 + CCM:
    - 14 FPS - power not measured
  - GPUISP v2 + CCM:
    - 30 FPS - sensor linerate - CPU usage ~ 70 % pulling 8 Watts.
- Texas Instruments AM69 - imx219
  - CPUISP + CCM
    - 4 FPS unknown power
  - GPUISP v1 + CCM
    - 2 or 3 FPS (doh)
  - GPUISP v2 + CCM
    - 15 FPS - linerate
- V2 series on-list
  - https://lists.libcamera.org/pipermail/libcamera-devel/2025-August/052550.html

Next steps

- Get GPUISP upstream
    - Lens shade correction
    - Lens distortion correction
    - Denoise
    - Dead Pixel correction
    - Lens flare correction
- Implement controls to declare features of CPU v GPU implementations
- Calibrate camera modules for better colors (per module CCMs)

Thank you !

Bedankt !

Go raibh maith agaibh!