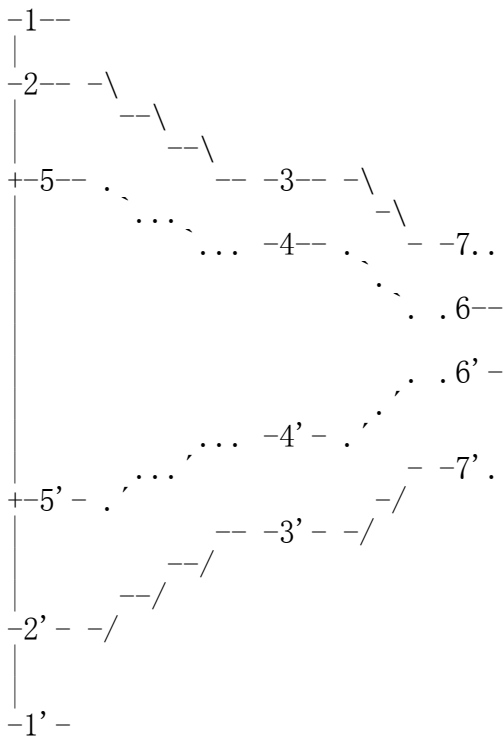


sh\_mobile\_ceu\_camera.txt.txt  
Cropping and Scaling algorithm, used in the sh\_mobile\_ceu\_camera driver  
=====

Terminology  
-----

sensor scales: horizontal and vertical scales, configured by the sensor driver  
host scales: -- host driver  
combined scales: sensor\_scale \* host\_scale

Generic scaling / cropping scheme  
-----



In the above chart minuses and slashes represent "real" data amounts, points and accents represent "useful" data, basically, CEU scaled and cropped output, mapped back onto the client's source plane.

Such a configuration can be produced by user requests:

S\_CROP(left / top = (5) - (1), width / height = (5') - (5))  
S\_FMT(width / height = (6') - (6))

Here:

- (1) to (1') - whole max width or height
- (1) to (2) - sensor cropped left or top
- (2) to (2') - sensor cropped width or height
- (3) to (3') - sensor scale
- (3) to (4) - CEU cropped left or top
- (4) to (4') - CEU cropped width or height
- (5) to (5') - reverse sensor scale applied to CEU cropped width or height

sh\_mobile\_ceu\_camera.txt.txt

- (2) to (5) - reverse sensor scale applied to CEU cropped left or top  
(6) to (6') - CEU scale - user window

S\_FMT

-----

Do not touch input rectangle - it is already optimal.

1. Calculate current sensor scales:

$$\text{scale}_s = ((3') - (3)) / ((2') - (2))$$

2. Calculate "effective" input crop (sensor subwindow) - CEU crop scaled back at current sensor scales onto input window - this is user S\_CROP:

$$\text{width}_u = (5') - (5) = ((4') - (4)) * \text{scale}_s$$

3. Calculate new combined scales from "effective" input window to requested user window:

$$\text{scale}_{\text{comb}} = \text{width}_u / ((6') - (6))$$

4. Calculate sensor output window by applying combined scales to real input window:

$$\text{width}_{s\_out} = ((2') - (2)) / \text{scale}_{\text{comb}}$$

5. Apply iterative sensor S\_FMT for sensor output window.

$$\text{subdev} \rightarrow \text{video\_ops} \rightarrow \text{s\_fmt}(.width = \text{width}_{s\_out})$$

6. Retrieve sensor output window (g\_fmt)

7. Calculate new sensor scales:

$$\text{scale}_{s\_new} = ((3')_{\text{new}} - (3)_{\text{new}}) / ((2') - (2))$$

8. Calculate new CEU crop - apply sensor scales to previously calculated "effective" crop:

$$\begin{aligned} \text{width}_{\text{ceu}} &= (4')_{\text{new}} - (4)_{\text{new}} = \text{width}_u / \text{scale}_{s\_new} \\ \text{left}_{\text{ceu}} &= (4)_{\text{new}} - (3)_{\text{new}} = ((5) - (2)) / \text{scale}_{s\_new} \end{aligned}$$

9. Use CEU cropping to crop to the new window:

$$\text{ceu\_crop}(.width = \text{width}_{\text{ceu}}, .left = \text{left}_{\text{ceu}})$$

10. Use CEU scaling to scale to the requested user window:

$$\text{scale}_{\text{ceu}} = \text{width}_{\text{ceu}} / \text{width}$$

S\_CROP

-----

sh\_mobile\_ceu\_camera.txt.txt

The API at <http://v4l2spec.bytesex.org/spec/x1904.htm> says:

"...specification does not define an origin or units. However by convention drivers should horizontally count unscaled samples relative to 0H."

We choose to follow the advise and interpret cropping units as client input pixels.

Cropping is performed in the following 6 steps:

1. Request exactly user rectangle from the sensor.
2. If smaller - iterate until a larger one is obtained. Result: sensor cropped to  $2 : 2'$ , target crop  $5 : 5'$ , current output format  $6' - 6$ .
3. In the previous step the sensor has tried to preserve its output frame as good as possible, but it could have changed. Retrieve it again.
4. Sensor scaled to  $3 : 3'$ . Sensor's scale is  $(2' - 2) / (3' - 3)$ . Calculate intermediate window:  $4' - 4 = (5' - 5) * (3' - 3) / (2' - 2)$
5. Calculate and apply host scale =  $(6' - 6) / (4' - 4)$
6. Calculate and apply host crop:  $6 - 7 = (5 - 2) * (6' - 6) / (5' - 5)$

--

Author: Guennadi Liakhovetski <g.liakhovetski@gmx.de>