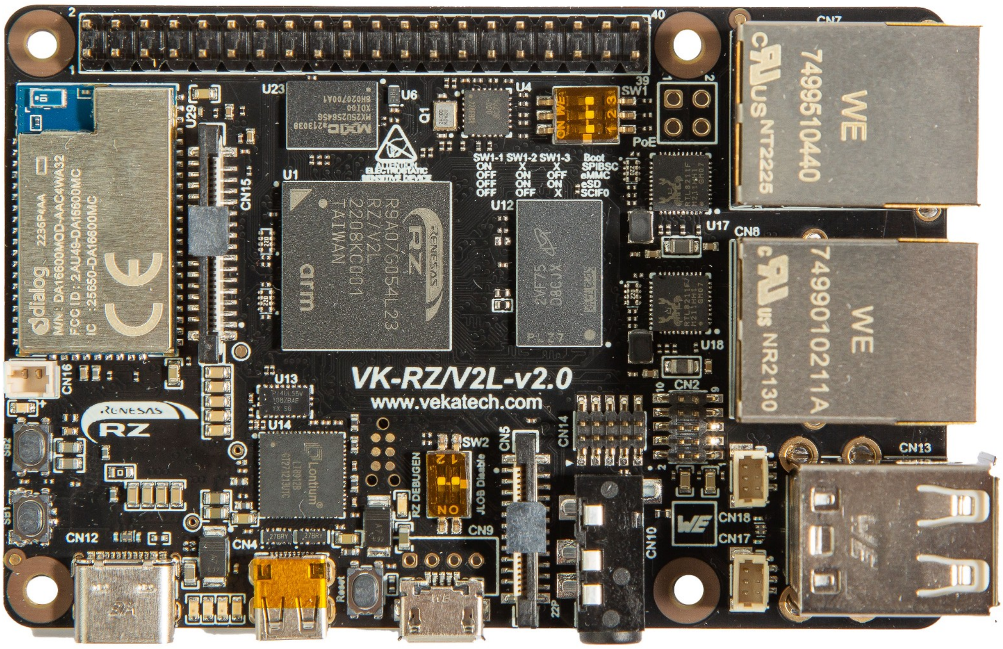
VK-RZ/V2L How To run OpenCV in Debian



VK-RZ/V2L v2.0 Board

Content:

1. Introduction

VK-RZ/V2L is industrial oriented board, compatible with Raspberry Pi 4 shields. It is based on Renesas **R9A07G054L23GBG**, **Dual ARM Cortex-A55 + Cortex-M33 MCU**. The main purpose of this manual is to show how to run OpenCV with DRP acceleration. For more info about this board, please read the full manual.

2. Solution

First, make sure you have running **Debian** OS on the **VK-RZ/V2L** board.

2.1 In case you have no OS . . .

That means you have completely empty MMC and you need to flash U-boot firmware. For more info how to do it, read the tool's manual, or follow the commands below:

- Connect **VK-RZ/V2L** to the PC (through **USB Type B micro**) & see what COM port is assigned by the OS in the Device Manager.
- Set the **SW1** of **VK-RZ/V2L** to (**1:OFF | 2:OFF |3:OFF**) so it can boot from **SCIF0**.
- Download **vkPyFlasher** rescue tool and unzip it to some folder of your choice.
- Unzip **uboot-vkrzv2l.zip** in: .<folder of your choice>/**vkPyFlasher/images**.
- Launch the tool: `cd vkPyFlasher`.

`flash_boot.py --board=vkrzv2l --serial_port=COM<n>`.

- Press **reset** button on the **VK-RZ/V2L**

2.2 In case you have OS, but it's not Debian . . .

That means you have Yocto or Home Assistant and you probably need to reflash the existing U-boot with the one that supports **DRP** (i.e. execute **2.1** first). If you have latest U-boot, you can proceed with **2.2** directly. To consider whether or not you have DRP support in U-boot, inspect it's environment variables. This can be done by opening **VK-RZ/V2L**'s COM port with 115200|8|N|1 settings, and pressing **reset** button on the board. Look carefully the log and when you see Hit any key to stop autoboot: x **hit** a key, type printenv and press **Enter**. If you spot a variable with name opencva you have DRP support and don't need to execute 2.1. After you are sure you have the right U-boot, you can proceed with the following commands:

So to flash Debian:

- Connect **VK-RZ/V2L** to the PC (through **USB Type B micro**) & see what COM port is assigned by the OS in the Device Manager.
- Connect **VK-RZ/V2L** to the PC (through **USB Type A device**) the **upper** one.
- Set the **SW1** of **VK-RZ/V2L** to (**1:OFF | 2:ON |3:OFF**) so it can boot from **eMMC**.
- Download **debian-bookworm-vkrzv2l.simg** in .../**vkPyFlasher/images/vkrzv2l**.
- Launch the tool: `cd vkPyFlasher`.

```
flash_img.py --board=vkrzv2l --serial_port=COM<n>--image_rootfs=debian-bookworm-vkrzv2l.simg.
```

- Press **reset** button on the **VK-RZ/V2L**
- Wait the flashing to complete.

2.3 In case you have Debian, install OpenCV v4.1.0 . . .

Once the Debian is booted, follow these commands:

- Get the executable binaries:

```
wget.https://vekatech.com/VK-RZ_V2L_docs/Demo/pkg/OpenCV/deb_opencv_4.1.0.sh.
```

- Make the file executable: `chmod +x deb_opencv_4.1.0.sh`.
- Install the library: `sudo ./deb_opencv_4.1.0.sh`.

2.4 In case you have OpenCV v4.1.0, run the sample . . .

You can see the beauty of the DRP module in action, by reducing the execution time for some of the OpenCV functions. Following the commands to see it:

- Make a folder for the OCA sample: `mkdir ~/OpenCVA && cd /OpenCVA`.
- Download the OCA sample:

```
wget https://vekatech.com/VK-RZ_V2L_docs/Demo/examples/Cpp/OpenCVA/deb_OCA_sample.
```

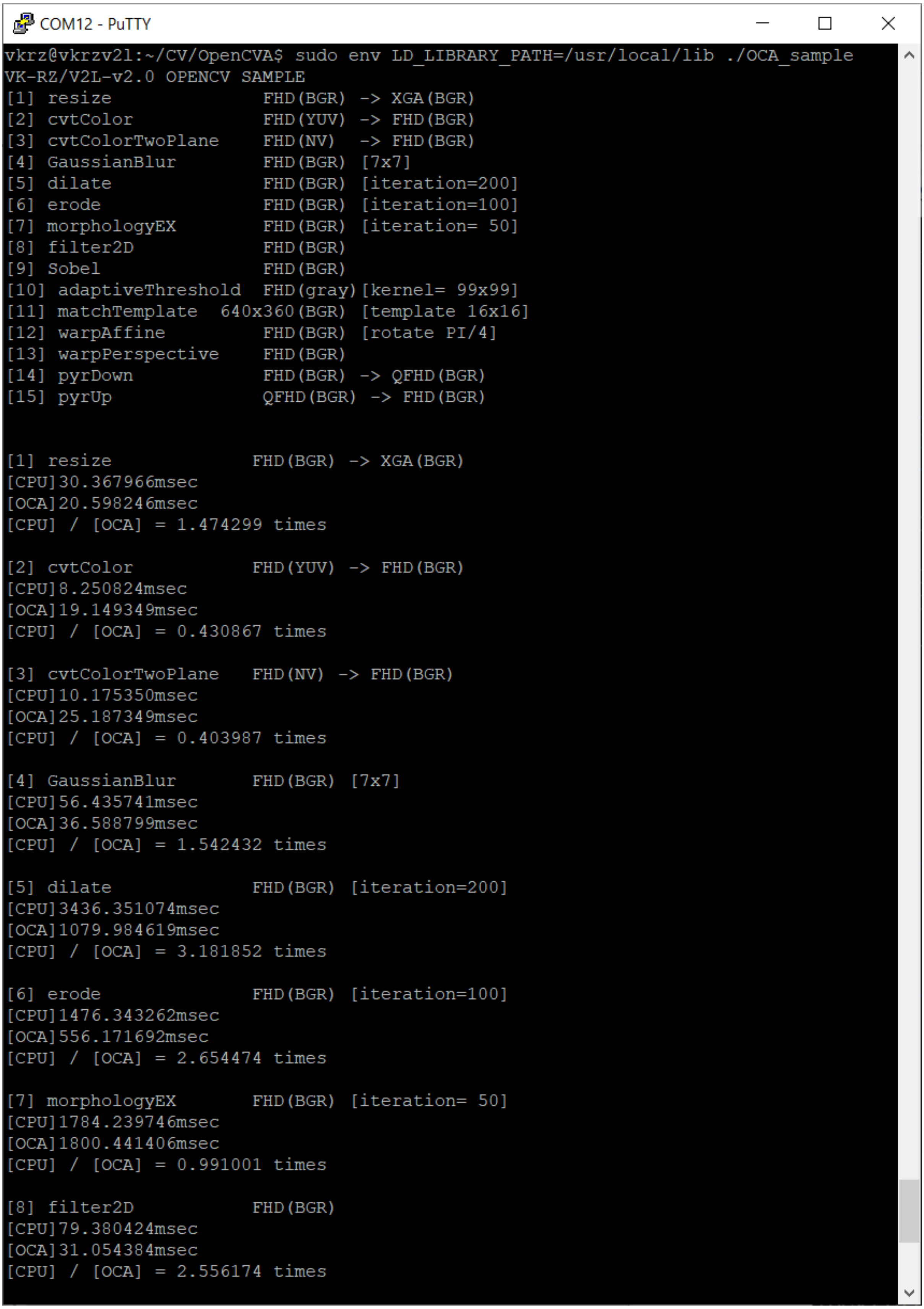
- Download the OCA input data:

```
wget https://vekatech.com/VK-RZ_V2L_docs/Demo/examples/Cpp/OpenCVA/image.png.
```

- Make the sample executable: `chmod +x deb_OCA_sample`.
- Execute the sample Cpp:

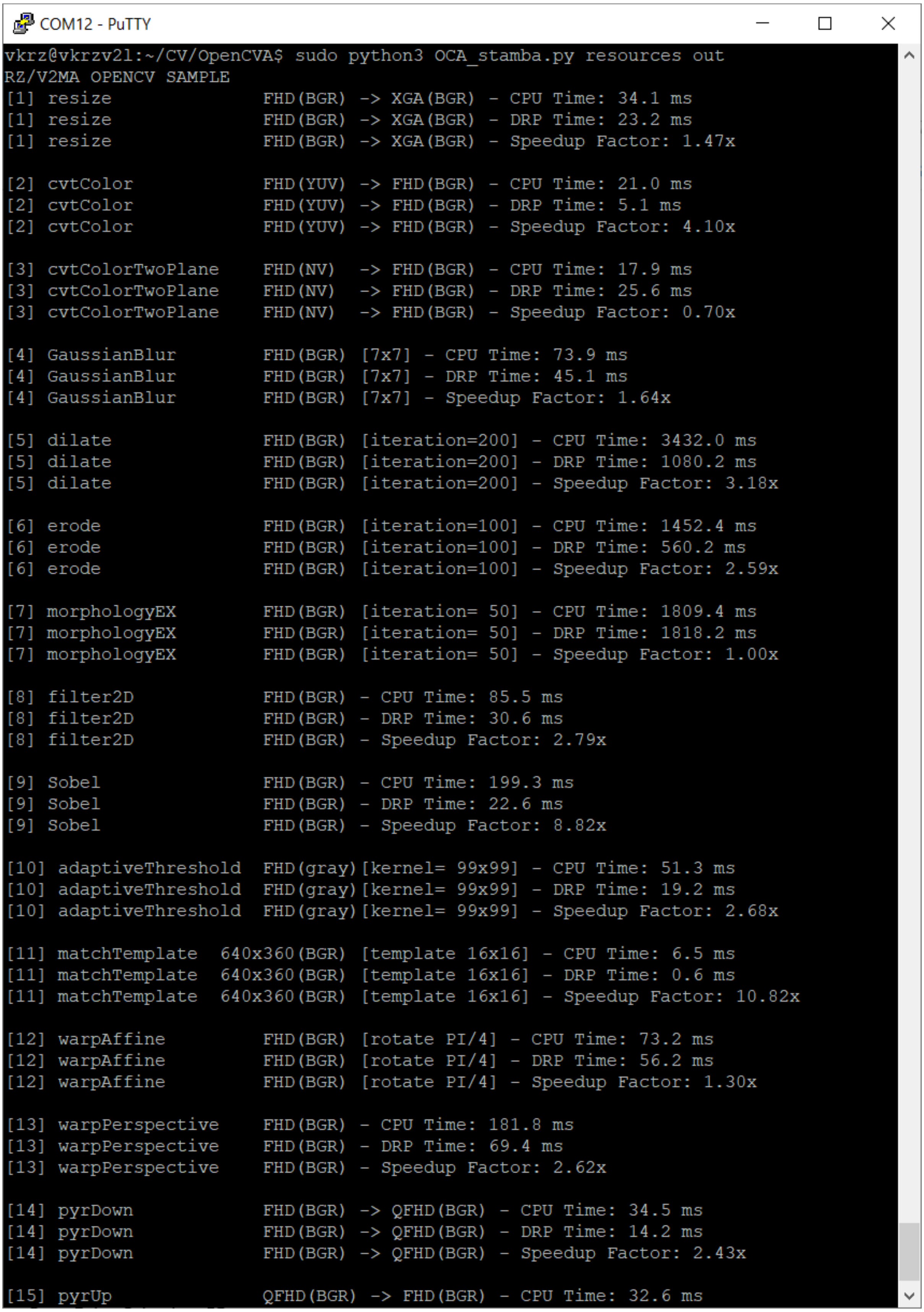
```
sudo env LD_LIBRARY_PATH=/usr/local/lib ./OCA_sample.
```

- Execute the sample Py: `sudo python3 OCA_sample.py recources out`.
- See the difference when DRP is used:



```
vkrz@vkrzv2l:~/CV/OpenCVA$ sudo env LD_LIBRARY_PATH=/usr/local/lib ./OCA_sample
VK-RZ/V2L-v2.0 OPENCV SAMPLE
[1] resize              FHD(BGR) -> XGA(BGR)
[2] cvtColor            FHD(YUV) -> FHD(BGR)
[3] cvtColorTwoPlane    FHD(NV)  -> FHD(BGR)
[4] GaussianBlur        FHD(BGR) [7x7]
[5] dilate              FHD(BGR) [iteration=200]
[6] erode               FHD(BGR) [iteration=100]
[7] morphologyEX        FHD(BGR) [iteration= 50]
[8] filter2D            FHD(BGR)
[9] Sobel               FHD(BGR)
[10] adaptiveThreshold  FHD(gray)[kernel= 99x99]
[11] matchTemplate  640x360(BGR) [template 16x16]
[12] warpAffine         FHD(BGR) [rotate PI/4]
[13] warpPerspective    FHD(BGR)
[14] pyrDown            FHD(BGR) -> QFHD(BGR)
[15] pyrUp              QFHD(BGR) -> FHD(BGR)


[1] resize             FHD(BGR) -> XGA(BGR)
[CPU]30.367966msec
[OCA]20.598246msec
[CPU] / [OCA] = 1.474299 times

[2] cvtColor           FHD(YUV) -> FHD(BGR)
[CPU]8.250824msec
[OCA]19.149349msec
[CPU] / [OCA] = 0.430867 times

[3] cvtColorTwoPlane   FHD(NV) -> FHD(BGR)
[CPU]10.175350msec
[OCA]25.187349msec
[CPU] / [OCA] = 0.403987 times

[4] GaussianBlur       FHD(BGR) [7x7]
[CPU]56.435741msec
[OCA]36.588799msec
[CPU] / [OCA] = 1.542432 times

[5] dilate             FHD(BGR) [iteration=200]
[CPU]3436.351074msec
[OCA]1079.984619msec
[CPU] / [OCA] = 3.181852 times

[6] erode              FHD(BGR) [iteration=100]
[CPU]1476.343262msec
[OCA]556.171692msec
[CPU] / [OCA] = 2.654474 times

[7] morphologyEX       FHD(BGR) [iteration= 50]
[CPU]1784.239746msec
[OCA]1800.441406msec
[CPU] / [OCA] = 0.991001 times

[8] filter2D           FHD(BGR)
[CPU]79.380424msec
[OCA]31.054384msec
[CPU] / [OCA] = 2.556174 times
```

with and without DRP acceleration (Cpp)



COM12 - PuTTY

```
vkrz@vkrzv2l:~/CV/OpenCVA$ sudo python3 OCA_stamba.py resources out
RZ/V2MA OPENCV SAMPLE
[1] resize              FHD(BGR) -> XGA(BGR) - CPU Time: 34.1 ms
[1] resize              FHD(BGR) -> XGA(BGR) - DRP Time: 23.2 ms
[1] resize              FHD(BGR) -> XGA(BGR) - Speedup Factor: 1.47x

[2] cvtColor            FHD(YUV) -> FHD(BGR) - CPU Time: 21.0 ms
[2] cvtColor            FHD(YUV) -> FHD(BGR) - DRP Time: 5.1 ms
[2] cvtColor            FHD(YUV) -> FHD(BGR) - Speedup Factor: 4.10x

[3] cvtColorTwoPlane    FHD(NV)  -> FHD(BGR) - CPU Time: 17.9 ms
[3] cvtColorTwoPlane    FHD(NV)  -> FHD(BGR) - DRP Time: 25.6 ms
[3] cvtColorTwoPlane    FHD(NV)  -> FHD(BGR) - Speedup Factor: 0.70x

[4] GaussianBlur        FHD(BGR) [7x7] - CPU Time: 73.9 ms
[4] GaussianBlur        FHD(BGR) [7x7] - DRP Time: 45.1 ms
[4] GaussianBlur        FHD(BGR) [7x7] - Speedup Factor: 1.64x

[5] dilate              FHD(BGR) [iteration=200] - CPU Time: 3432.0 ms
[5] dilate              FHD(BGR) [iteration=200] - DRP Time: 1080.2 ms
[5] dilate              FHD(BGR) [iteration=200] - Speedup Factor: 3.18x

[6] erode               FHD(BGR) [iteration=100] - CPU Time: 1452.4 ms
[6] erode               FHD(BGR) [iteration=100] - DRP Time: 560.2 ms
[6] erode               FHD(BGR) [iteration=100] - Speedup Factor: 2.59x

[7] morphologyEX        FHD(BGR) [iteration= 50] - CPU Time: 1809.4 ms
[7] morphologyEX        FHD(BGR) [iteration= 50] - DRP Time: 1818.2 ms
[7] morphologyEX        FHD(BGR) [iteration= 50] - Speedup Factor: 1.00x

[8] filter2D            FHD(BGR) - CPU Time: 85.5 ms
[8] filter2D            FHD(BGR) - DRP Time: 30.6 ms
[8] filter2D            FHD(BGR) - Speedup Factor: 2.79x

[9] Sobel               FHD(BGR) - CPU Time: 199.3 ms
[9] Sobel               FHD(BGR) - DRP Time: 22.6 ms
[9] Sobel               FHD(BGR) - Speedup Factor: 8.82x

[10] adaptiveThreshold  FHD(gray)[kernel= 99x99] - CPU Time: 51.3 ms
[10] adaptiveThreshold  FHD(gray)[kernel= 99x99] - DRP Time: 19.2 ms
[10] adaptiveThreshold  FHD(gray)[kernel= 99x99] - Speedup Factor: 2.68x

[11] matchTemplate  640x360(BGR) [template 16x16] - CPU Time: 6.5 ms
[11] matchTemplate  640x360(BGR) [template 16x16] - DRP Time: 0.6 ms
[11] matchTemplate  640x360(BGR) [template 16x16] - Speedup Factor: 10.82x

[12] warpAffine         FHD(BGR) [rotate PI/4] - CPU Time: 73.2 ms
[12] warpAffine         FHD(BGR) [rotate PI/4] - DRP Time: 56.2 ms
[12] warpAffine         FHD(BGR) [rotate PI/4] - Speedup Factor: 1.30x

[13] warpPerspective    FHD(BGR) - CPU Time: 181.8 ms
[13] warpPerspective    FHD(BGR) - DRP Time: 69.4 ms
[13] warpPerspective    FHD(BGR) - Speedup Factor: 2.62x

[14] pyrDown            FHD(BGR) -> QFHD(BGR) - CPU Time: 34.5 ms
[14] pyrDown            FHD(BGR) -> QFHD(BGR) - DRP Time: 14.2 ms
[14] pyrDown            FHD(BGR) -> QFHD(BGR) - Speedup Factor: 2.43x

[15] pyrUp              QFHD(BGR) -> FHD(BGR) - CPU Time: 32.6 ms
```

with and without DRP acceleration (Py)

Revision overview list

Revision number	Description changes
0.1	Initial