moments_PL

February 12, 2020

1 Moment calculator in Programmable Logic (PL) - Application Notebook

This reference design illustrates how to run the calculator and resizer IP on the Programmable Logic (PL) using Jupyter Notebooks and Python

> The moments IP calculates the moments of a blob. Its center and orientation is returned. Also the image is resized to half size.
>
> https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_1/ug1233-xilinx-opencv-user-guide.pdf

1.1 Contents

1.2 Image Moments in Programmable Logic

1.3 Import libraries

[83]:
```
from PIL import Image, ImageDraw
import numpy as np
from IPython.display import display
from pynq import Xlnk
from pynq import Overlay
import math # for atan2
```

1.4 Download the Moments IP bitstream

[84]:
```
moments_design = Overlay("../bitstream/moments.bit")
#moments_design?
```

Then create DMA and Moments IP objects.

[85]:
```
dma = moments_design.axi_dma_0
moments = moments_design.moments_0
```

1.5 Create an Image object using PIL in SW

We will load image from the SD card and create a PIL Image object.

[138]:
```
#image_path = "simu_img/Block5.png"
#image_path = "simu_img/Block7.png"
#image_path = "simu_img/eye1.jpg"
#image_path = "simu_img/test0.png"
image_path = "simu_img/test2.png"
original_image = Image.open(image_path)
original_image.load()
```

[138]:
```
<PixelAccess at 0xadb6b780>
```

We now create a numpy array of the pixels.

[139]:
```
input_array = np.array(original_image)
```

1.6 Display the image to calculate moments and center from

[140]:
```
input_image = Image.fromarray(input_array)
display(input_image)
```

[140]:

Let's double-check the original color image size.

[141]:
```
old_width, old_height = original_image.size
print("Image size: {}x{} pixels.".format(old_width, old_height))
```

```
Image size: 640x360 pixels.
```

1.7 Calculate

We can set new dimensions for resizing. 1. Downscale factor range: 2 (by design of the resize IP) 2. Input size is 640x360 and output size of Image is 320x180 for this version of resize IP

[142]:
```
new_width = int(old_width/2)
new_height = int(old_height/2)
```

We now allocate memory to process data on PL. Data is provided as contiguous memory blocks. The size of the buffer depends on the size of the input or output data. The image dimensions extracted from the read image are used to allocate contiguous memory blocks.
We will call cma_array() to perform the allocation.

[143]:
```
xlnk = Xlnk()
in_buffer = xlnk.cma_array(shape=(old_height, old_width, 3),
                           dtype=np.uint8, cacheable=1)
out_buffer = xlnk.cma_array(shape=(new_height, new_width, 3),
                            dtype=np.uint8, cacheable=1)
```

Documentation snippet for xlnk.cma_array:

```
"""
Get a contiguously allocated numpy array

Parameters
----------
shape : int or tuple of int
    The dimensions of the array to construct - We use (height, width, depth)
dtype : numpy.dtype or str
    The data type to construct - We use 8-bit unsigned int

"""
```

We now display the image in buffer.

Note : The input_array has to be copied into the contiguous memory array (deep copy).

The input buffer size should be 640 x 360 x 3 (height x width x depth).

[144]:
```
if (len(input_array.shape)==2):
    # if single channel create rgb image 3channels
    h,w = input_array.shape
    rgbArray = np.zeros((h,w,3), 'uint8')
    rgbArray[:,:, 0] = input_array
    rgbArray[:,:, 1] = input_array
    rgbArray[:,:, 2] = input_array
    in_buffer[0:640*360*3] = rgbArray
else:
    in_buffer[0:640*360*3] = input_array

buf_image = Image.fromarray(in_buffer)

display(buf_image)
print("Color image size: {}x{} pixels.".format(old_width, old_height))
```

[144]:

Color image size: 640x360 pixels.

We can now run the resizer IP. We will push the data from input buffer through the pipeline to the output buffer.

We will first need to setup moments input, output and DMA IPs using MMIO interface before we stream image data to them. For example, we can write one value to MMIO registers of resizer and read three.

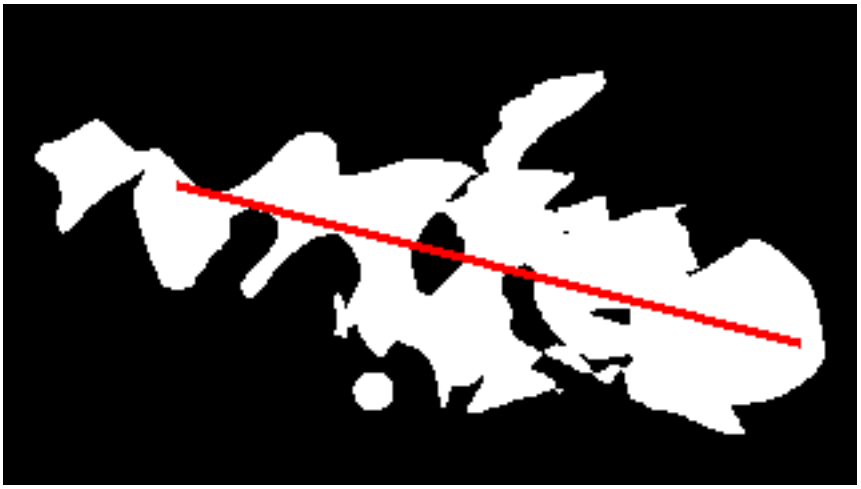
register offset	configuration
0x10	number (input extension)
0x18	number out x center
0x20	number out y center
0x28	number out angle x componet
0x30	number out angle y componet

[145]:
```
for i in range(2):
    moments.write(0x10, 13)
    x = moments.read(0x18)
    y = moments.read(0x20)
    anglex = moments.read(0x28)
    angley = moments.read(0x30)

    def run_kernel():
        dma.sendchannel.transfer(in_buffer)
        dma.recvchannel.transfer(out_buffer)
```

```
        moments.write(0x00,0x81) # start
        dma.sendchannel.wait()
        dma.recvchannel.wait()

    run_kernel()

    result = Image.fromarray(out_buffer)

    if  angley & 0x80000000:
        angley -= 4294967295
    if  anglex & 0x80000000:
        anglex -= 4294967295

    angleRAD = 0.5 * math.atan2(angley,anglex)

print('Return angle components: angle x comp: {} angle y comp: {}'.
 ↪format(anglex, angley))
print('Return values of moments: x: {} y: {} angleRAD: {}'.format(x,y,angleRAD))
print("Image has also been resized in Hardware(PL): {}x{} pixels.".
 ↪format(new_width, new_height))
ll = 120
draw = ImageDraw.Draw(result)
draw.line((x-ll*np.cos(-angleRAD),y+ll*np.sin(-angleRAD),x+ll*np.
 ↪cos(-angleRAD),y-ll*np.sin(-angleRAD)), fill=255, width=4)
del draw
display(result)
```

```
Return angle components: angle x comp: 4497 angle y comp: 2472
Return values of moments: x: 182 y: 97 angleRAD: 0.25130635088140046
Image has also been resized in Hardware(PL): 320x180 pixels.
```

[145]:



1.8 We can time the moment calculator and resizing in PL operation

[146]:
```
%%timeit

moments.write(0x10, 13)
moments.read(0x18)
moments.read(0x20)
moments.read(0x28)
moments.read(0x30)

def run_kernel():
    dma.sendchannel.transfer(in_buffer)
    dma.recvchannel.transfer(out_buffer)
    moments.write(0x00,0x81) # start
    dma.sendchannel.wait()
    dma.recvchannel.wait()

run_kernel()

result = Image.fromarray(out_buffer)
```

```
100 loops, best of 3: 11.5 ms per loop
```

Finally we need to reset all the contiguous memory buffers.

[147]:
```
xlnk.xlnk_reset()
```

Section 1.1

2 References

Image from original resizer_PL notebook, change to reflect its new behavior as moment calculator and resizer.

https://pillow.readthedocs.io/en/3.1.x/index.html
https://github.com/Xilinx/PYNQ/blob/master/docs/source/python_environment.ipynb
https://github.com/Xilinx/PYNQ/blob/master/docs/source/jupyter_notebooks.ipynb
https://github.com/Xilinx/PYNQ/blob/master/docs/source/jupyter_notebooks_advanced_features.ipy

[0]:
```

```