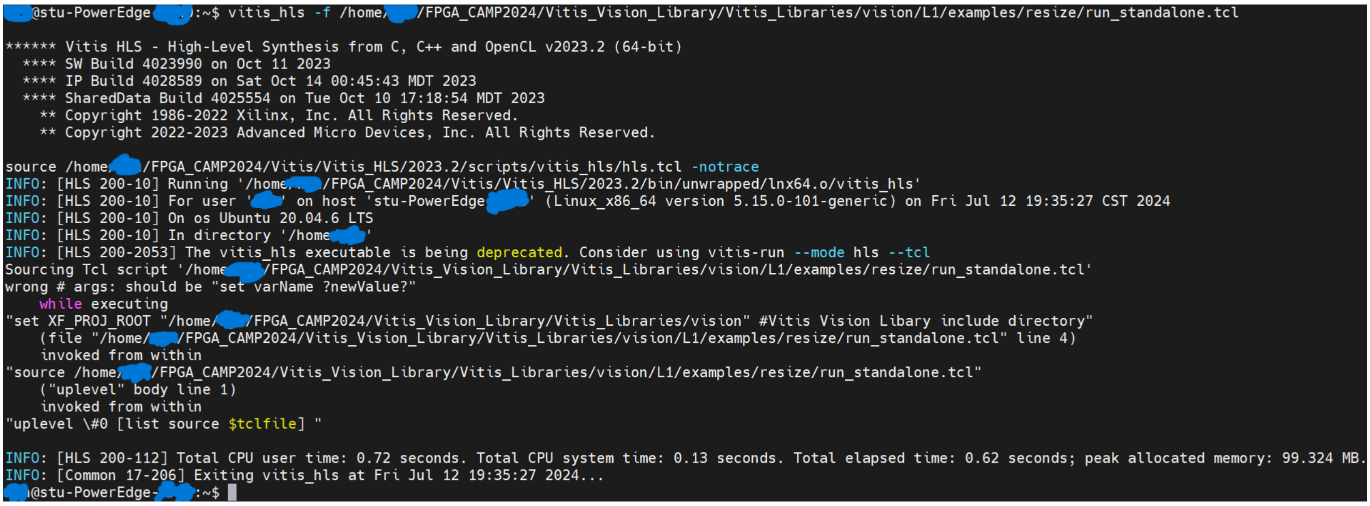
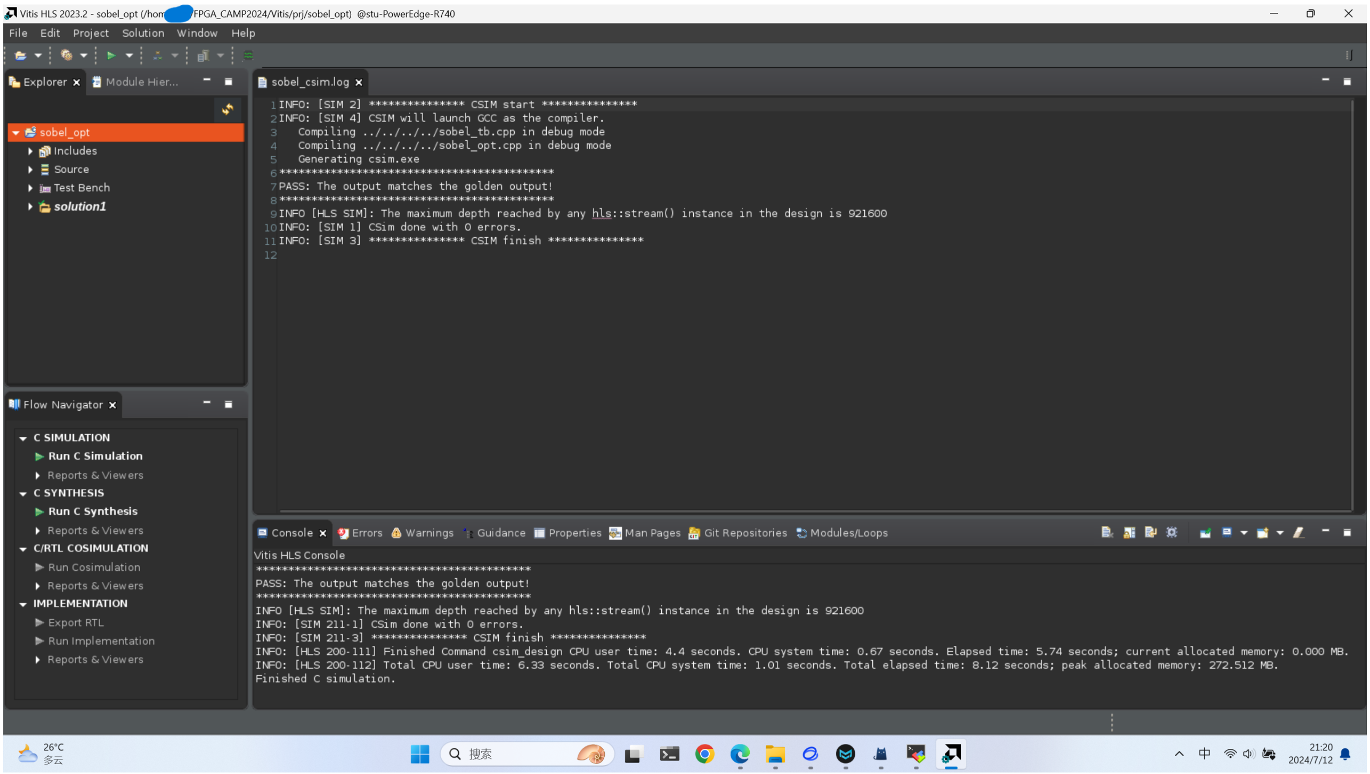
lab1完成情况：

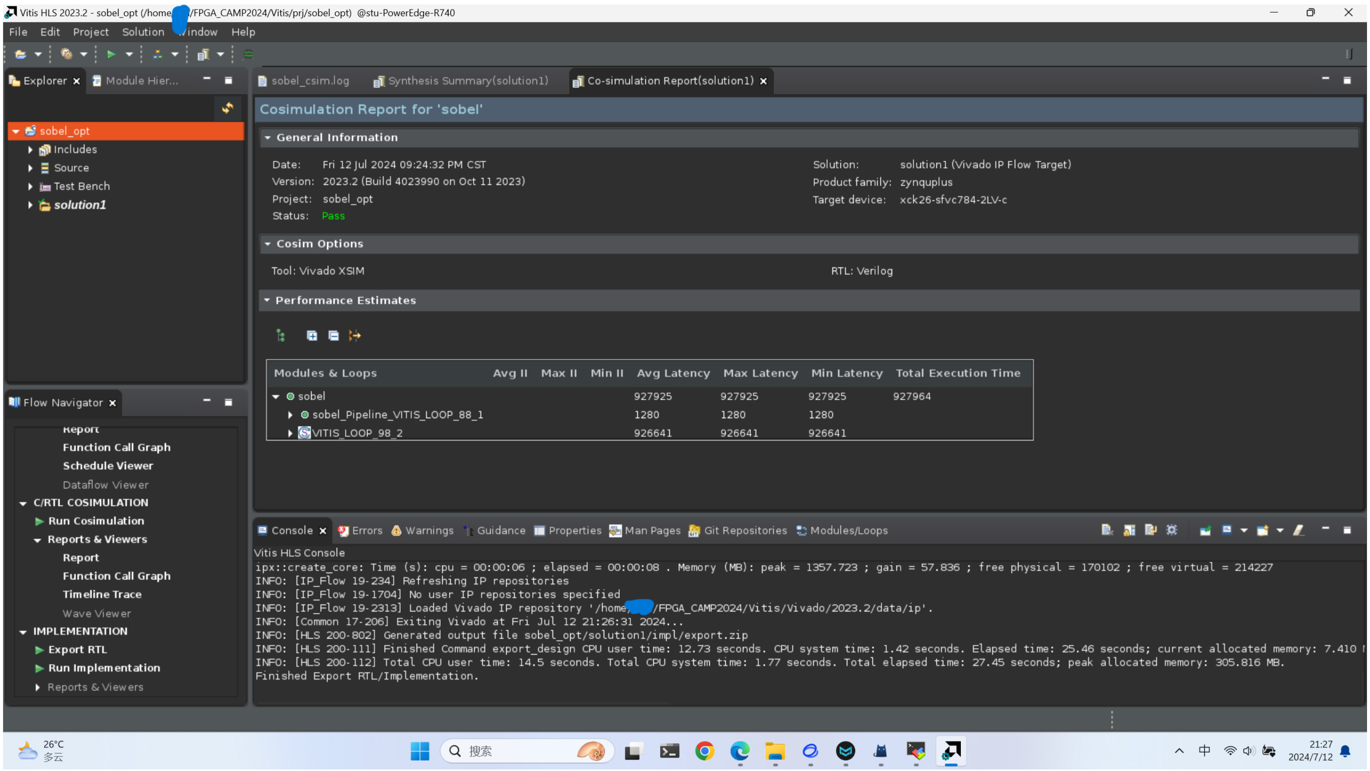
1、根据sobel gui flow完成了Linux环境的配置。按流程完成了sobel ip的综合实现与bit、hwh文件的生成。

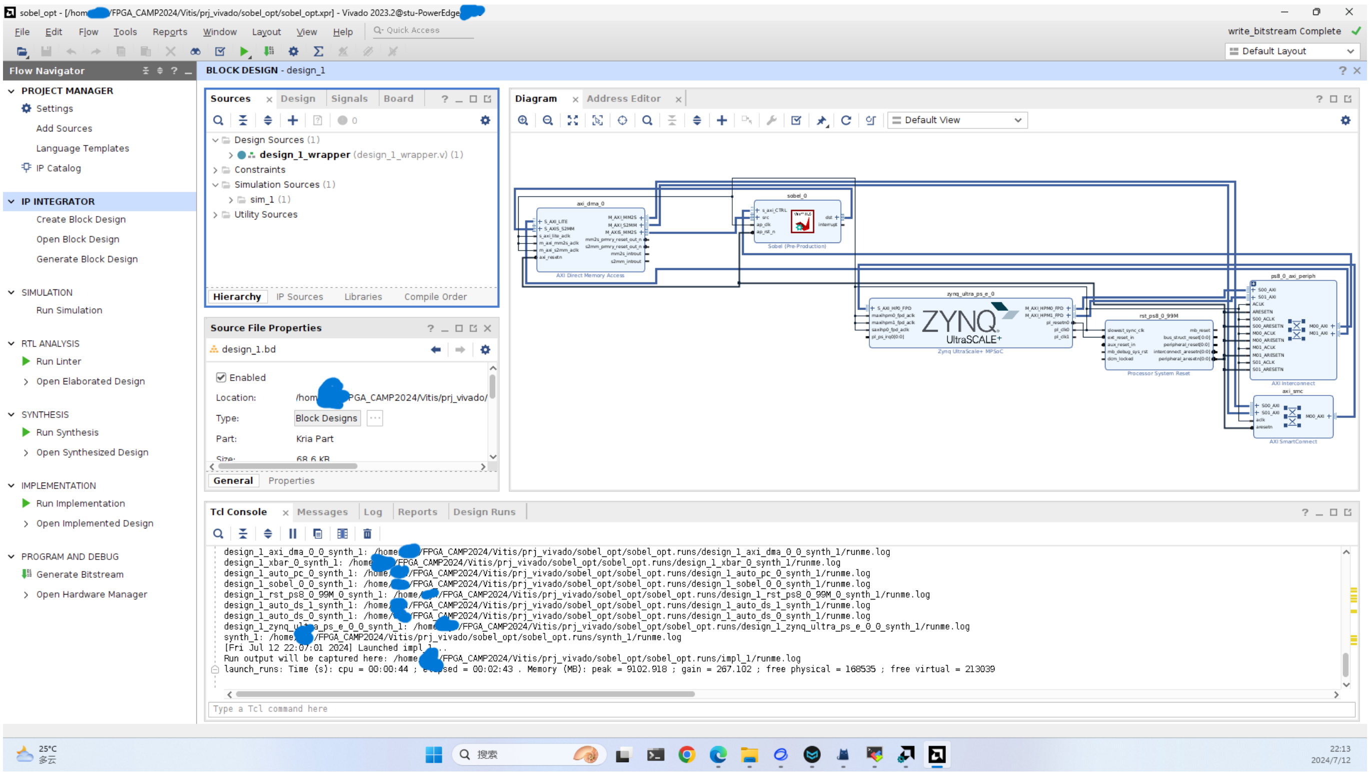
2、完成尝试调整图片尺寸输入，完成了图像处理的基本流程。

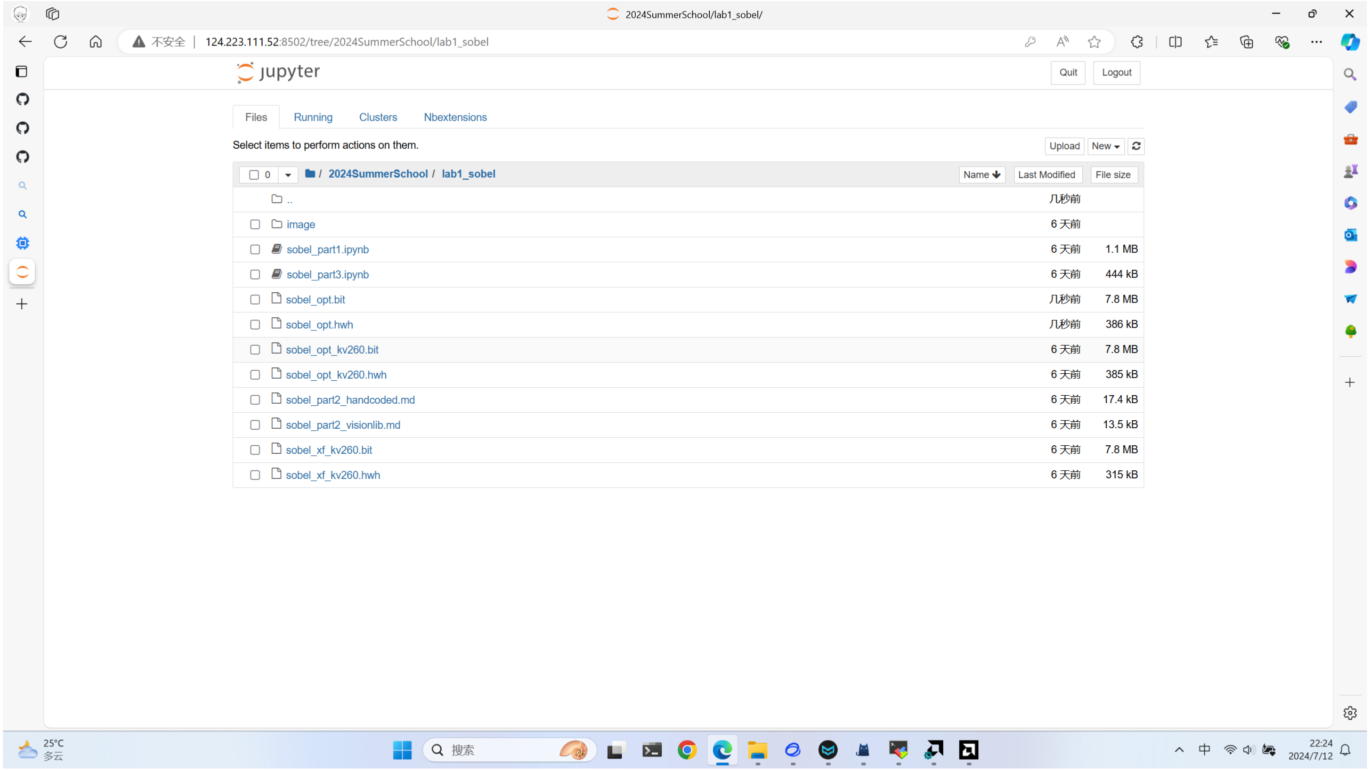
1：






Vitis HLS 2023.2 - sobel_opt (/home[redacted]/FPGA_CAMP2024/Vitis/prj/sobel_opt) @stu-PowerEdge-R740
File Edit Project Solution [redacted]indow Help
Explorer Module Hier...
sobel_opt
Includes
Source
Test Bench
solution1
sobel_csim.log Synthesis Summary(solution1) Co-simulation Report(solution1)
Cosimulation Report for 'sobel'
General Information
Date: Fri 12 Jul 2024 09:24:32 PM CST
Version: 2023.2 (Build 4023990 on Oct 11 2023)
Project: sobel_opt
Status: Pass
Solution: solution1 (Vivado IP Flow Target)
Product family: zynquplus
Target device: xck26-sfvc784-2LV-c
Cosim Options
Tool: Vivado XSIM
RTL: Verilog
Performance Estimates
Modules & Loops | Avg II | Max II | Min II | Avg Latency | Max Latency | Min Latency | Total Execution Time
sobel | | | | 927925 | 927925 | 927925 | 927964
sobel_Pipeline_VITIS_LOOP_88_1 | | | | 1280 | 1280 | 1280
VITIS_LOOP_98_2 | | | | 926641 | 926641 | 926641
Flow Navigator
Report
Function Call Graph
Schedule Viewer
Dataflow Viewer
C/RTL COSIMULATION
Run Cosimulation
Reports & Viewers
Report
Function Call Graph
Timeline Trace
Wave Viewer
IMPLEMENTATION
Export RTL
Run Implementation
Reports & Viewers
Console Errors Warnings Guidance Properties Man Pages Git Repositories Modules/Loops
Vitis HLS Console
ipx::create_core: Time (s): cpu = 00:00:06 ; elapsed = 00:00:08 . Memory (MB): peak = 1357.723 ; gain = 57.836 ; free physical = 170102 ; free virtual = 214227
INFO: [IP_Flow 19-234] Refreshing IP repositories
INFO: [IP_Flow 19-1704] No user IP repositories specified
INFO: [IP_Flow 19-2313] Loaded Vivado IP repository '/home[redacted]/FPGA_CAMP2024/Vitis/Vivado/2023.2/data/ip'.
INFO: [Common 17-206] Exiting Vivado at Fri Jul 12 21:26:31 2024...
INFO: [HLS 200-802] Generated output file sobel_opt/solution1/impl/export.zip
INFO: [HLS 200-111] Finished Command export_design CPU user time: 12.73 seconds. CPU system time: 1.42 seconds. Elapsed time: 25.46 seconds; current allocated memory: 7.410
INFO: [HLS 200-112] Total CPU user time: 14.5 seconds. Total CPU system time: 1.77 seconds. Total elapsed time: 27.45 seconds; peak allocated memory: 305.816 MB.
Finished Export RTL/Implementation.
26°C 多云
搜索
21:27 2024/7/12


sobel_opt - [/hom[redacted]/FPGA_CAMP2024/Vitis/prj_vivado/sobel_opt/sobel_opt.xpr] - Vivado 2023.2@stu-PowerEdge[redacted]
File Edit Flow Tools Reports Window Layout View Help Quick Access
write_bitstream Complete
Default Layout
Flow Navigator
BLOCK DESIGN - design_1
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Run Linter
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
Sources Design Signals Board
Design Sources (1)
design_1_wrapper (design_1_wrapper.v) (1)
Constraints
Simulation Sources (1)
sim_1 (1)
Utility Sources
Hierarchy IP Sources Libraries Compile Order
Source File Properties
design_1.bd
Enabled
Location: /hom[redacted]PGA_CAMP2024/Vitis/prj_vivado/
Type: Block Designs
Part: Kria Part
Size: 68.6 kB
General Properties
Diagram Address Editor
Default View
axi_dma_0
AXI Direct Memory Access
sobel_0
Sobel (Pre-Production)
zynq_ultra_ps_e_0
ZYNQ UltraSCALE+
Zynq UltraScale+ MPSoC
rst_ps8_0_99M
Processor System Reset
ps8_0_axi_periph
AXI Interconnect
axi_smc
AXI SmartConnect
Tcl Console Messages Log Reports Design Runs
design_1_axi_dma_0_0_synth_1: /home[redacted]/FPGA_CAMP2024/Vitis/prj_vivado/sobel_opt/sobel_opt.runs/design_1_axi_dma_0_0_synth_1/runme.log
design_1_xbar_0_synth_1: /home[redacted]FPGA_CAMP2024/Vitis/prj_vivado/sobel_opt/sobel_opt.runs/design_1_xbar_0_synth_1/runme.log
design_1_auto_pc_0_synth_1: /home[redacted]/FPGA_CAMP2024/Vitis/prj_vivado/sobel_opt/sobel_opt.runs/design_1_auto_pc_0_synth_1/runme.log
design_1_sobel_0_0_synth_1: /home[redacted]/FPGA_CAMP2024/Vitis/prj_vivado/sobel_opt/sobel_opt.runs/design_1_sobel_0_0_synth_1/runme.log
design_1_rst_ps8_0_99M_0_synth_1: /home[redacted]/FPGA_CAMP2024/Vitis/prj_vivado/sobel_opt/sobel_opt.runs/design_1_rst_ps8_0_99M_0_synth_1/runme.log
design_1_auto_ds_1_synth_1: /home[redacted]/FPGA_CAMP2024/Vitis/prj_vivado/sobel_opt/sobel_opt.runs/design_1_auto_ds_1_synth_1/runme.log
design_1_auto_ds_0_synth_1: /home[redacted]FPGA_CAMP2024/Vitis/prj_vivado/sobel_opt/sobel_opt.runs/design_1_auto_ds_0_synth_1/runme.log
design_1_zynq_u[redacted]a_ps_e_0_0_synth_1: /home[redacted]/FPGA_CAMP2024/Vitis/prj_vivado/sobel_opt/sobel_opt.runs/design_1_zynq_ultra_ps_e_0_0_synth_1/runme.log
synth_1: /home[redacted]/FPGA_CAMP2024/Vitis/prj_vivado/sobel_opt/sobel_opt.runs/synth_1/runme.log
[Fri Jul 12 22:07:01 2024] Launched impl_1...
Run output will be captured here: /home[redacted]/FPGA_CAMP2024/Vitis/prj_vivado/sobel_opt/sobel_opt.runs/impl_1/runme.log
launch_runs: Time (s): cpu = 00:00:44 ; e[redacted]sed = 00:02:43 . Memory (MB): peak = 9102.918 ; gain = 267.102 ; free physical = 168535 ; free virtual = 213039
Type a Tcl command here
25°C 多云
搜索
22:13 2024/7/12



2：

2.1 Read the Image in JPG Format

In [2]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
img = cv2.imread("./image/cat.jpg")
print("original image size: {}".format(img.shape))
plt.imshow(img[:, :, ::-1])
```

original image size: (1200, 1920, 3)

Out[2]: <matplotlib.image.AxesImage at 0xa2761eb0>

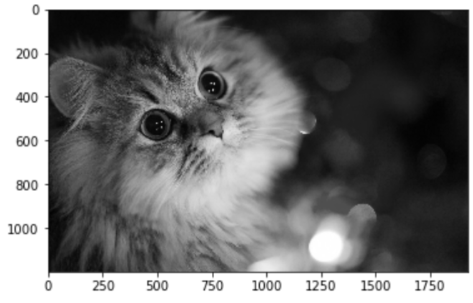
2.2 Converting RGB to Grayscale

In the following, we convert the RGB image into a grayscale map to visualize the change of data dimension.

In [3]:
```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
print("gray image size: {}".format(gray.shape))
plt.imshow(gray,cmap='gray')
```

```
gray image size: (1200, 1920)
```

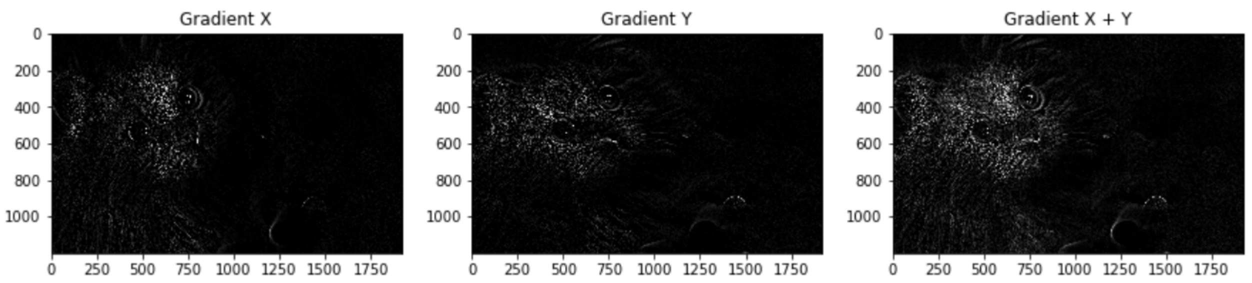
Out[3]:
```
<matplotlib.image.AxesImage at 0xa2590358>
```



In [4]:
```
import time
start_time = time.time()
sobel_x =cv2.Sobel(gray,cv2.CV_8U ,1,0)
sobel_y = cv2.Sobel(gray,cv2.CV_8U ,0,1)
sobel_res = np.clip(sobel_x + sobel_y, 0, 255)
end_time = time.time()
print("Time cost with Python: {}s".format(end_time - start_time))
fig_sobel3 = plt.figure()
fig_sobel3.set_figheight(4)
fig_sobel3.set_figwidth(15)

# gradient x
fig_1 = fig_sobel3.add_subplot(131)
fig_1.title.set_text('Gradient X')
plt.imshow(sobel_x,cmap='gray')

# gradient y
fig_2 = fig_sobel3.add_subplot(132)
fig_2.title.set_text('Gradient Y')
plt.imshow(sobel_y,cmap='gray')

# gradient
fig_3 = fig_sobel3.add_subplot(133)
fig_3.title.set_text('Gradient X + Y')
plt.imshow(sobel_res,cmap='gray')
```

```
Time cost with Python: 0.44332861900329593s
```

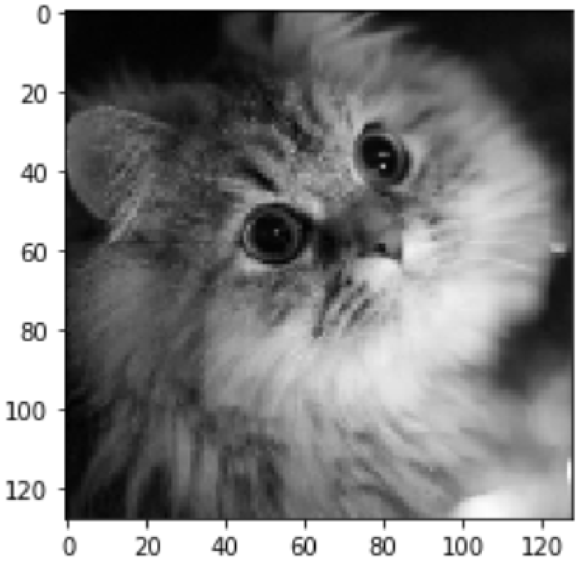
Out[4]:
```
<matplotlib.image.AxesImage at 0xa2115ca0>
```



2.1.1 Load the Original Grayscale Image

In [1]:
```
import cv2 as cv
from matplotlib import pyplot as plt
import numpy as np
import time
import copy
gray = cv.imread("./image/cat1200x1200_128x128.png", cv.IMREAD_GRAYSCALE)
plt.imshow(gray, cmap="gray")
```

Out[1]:
```
<matplotlib.image.AxesImage at 0xa27ffd90>
```



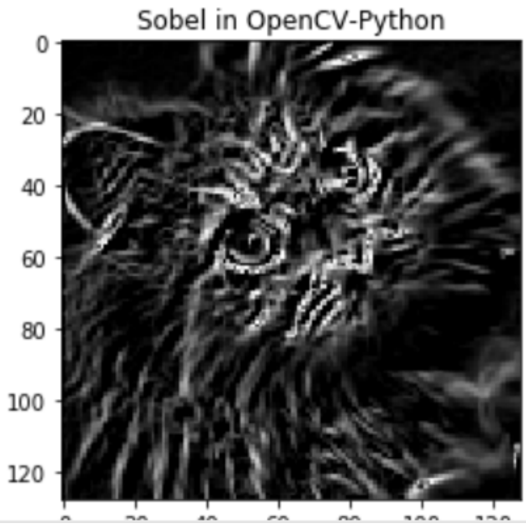
2.1.2 Using `cv.Sobel()`

In [2]:

```
# Apply Sobel filter to get the gradients in X and Y direction
# The Sobel operator calculates the gradient of the image intensity at each pixel

start_time_python = time.time()

Gx = cv.Sobel(gray, cv.CV_16U, 1, 0, ksize=3)  # Gradient in X direction
Gy = cv.Sobel(gray, cv.CV_16U, 0, 1, ksize=3)  # Gradient in Y direction

# Compute the gradient magnitude
G = np.sqrt(Gx**2 + Gy**2)
# Convert to 8-bit image for display
G_python = G.astype(np.uint8)

end_time_python = time.time()

time_python = end_time_python - start_time_python

print("Time cost with software: {}s".format(time_python))

plt.title("Sobel in OpenCV-Python")
plt.imshow(G_python, cmap="gray")
```

```
Time cost with software: 0.010274171829223633s
```

Out[2]: `<matplotlib.image.AxesImage at 0xa26a6b08>`



In [6]:

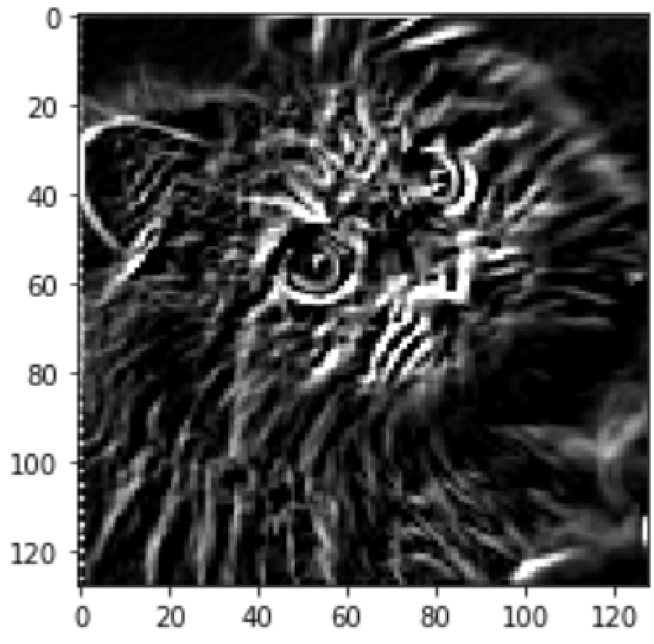
```
dma = overlay.axi_dma_0
```

In [7]:

```
import time

sobel.register_map.CTRL.AP_START = 1

start_time = time.time()

dma.sendchannel.transfer(input_buffer)
dma.recvchannel.transfer(output_buffer)
dma.sendchannel.wait() # wait for send channel
dma.recvchannel.wait() # wait for recv channel

end_time = time.time()

time_handcoded = end_time - start_time
print("Time cost with handcoded IP: {}s".format(time_handcoded))
# G_handcoded = copy.deepcopy(output_buffer.reshape(rows, cols))
G_handcoded = output_buffer.reshape(rows, cols)
plt.imshow(G_handcoded,cmap='gray')
```

```
Time cost with handcoded IP: 0.0036556720733642578s
```

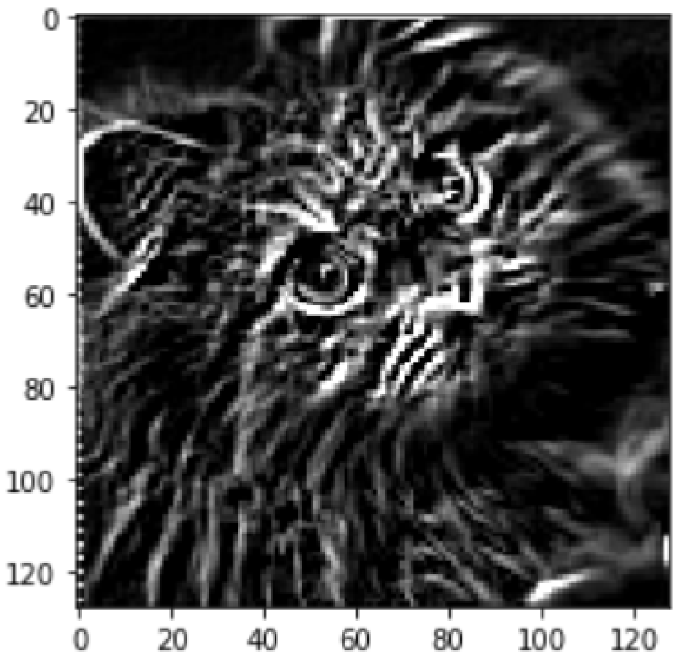
Out[7]: <matplotlib.image.AxesImage at 0xa1cb6dd8>



2.2.5 Reload the overlay to apply the changes

In [9]:
```
overlay = Overlay("./sobel_opt_z2.bit")
dma = overlay.axi_dma_0
sobel = overlay.sobel_0

res = sobel.sobel3x3(input_buffer, output_buffer, rows, cols)

plt.imshow(res,cmap='gray')
```

Out[9]: <matplotlib.image.AxesImage at 0xa1f859d0>



2.3.4 Starting IP

The control signal is located at address 0x00, which we can write and read to control whether the IP start and listen is completed.

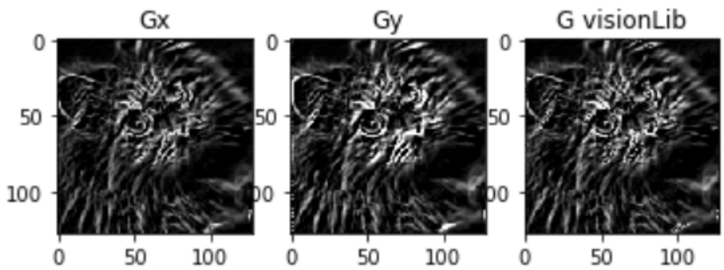
n [14]:
```
import time

sobel.write(0x00, 0x01)
start_time = time.time()
while True:
    reg = sobel.read(0x00)
    if reg != 1:
        break
end_time = time.time()
time_visionLib = end_time - start_time

print("Time cost with vison library IP: {}s".format(end_time - start_time))
```

```
Time cost with vison library IP: 0.0010945796966552734s
```

The results have been written to the `output_buffer`, which we can view.

In [15]:
```
G_visionLib = output_buffer_1.reshape(rows, cols) + output_buffer_2.reshape(rows, cols)
plt.subplot(131)
plt.title("Gx")
plt.imshow(G_python, cmap="gray")
plt.subplot(132)
plt.title("Gy")
plt.imshow(G_handcoded, cmap="gray")
plt.subplot(133)
plt.title("G visionLib")
plt.imshow(G_visionLib, cmap="gray")
```

Out[15]:
```
<matplotlib.image.AxesImage at 0xa1c6fb80>
```



We can simply load the hand-Coded and vision library version of Sobel to compare the performance.

In [16]:
```
import numpy as np
import matplotlib.pyplot as plt
import random

# prepare data
x_data = ['Sobel OpenCV-Python','Sobel Handwritten', 'Sobel Vision Library']
y_data = [time_python, time_handcoded, time_visionLib]


for i in range(len(x_data)):
    plt.bar(x_data[i], y_data[i])

for a,b in zip(x_data,y_data):
    plt.text(a,b,'%.4f'%b,ha='center',va='bottom',fontsize=11);

plt.title("Time used of different types")
plt.xlabel("Type")
plt.ylabel("Time(s)")

plt.show()
```