Hign Perfomance Computing. Домашнее задание 6

In [ ]:
```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

Выбрать файлы Файл не выбран

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving pic.jpg to pic.jpg
User uploaded file "pic.jpg" with length 249081 bytes
```

In [ ]:
```
!ls -la
```

```
total 260
drwxr-xr-x 1 root root   4096 Dec 11 19:30 .
drwxr-xr-x 1 root root   4096 Dec 11 19:19 ..
drwxr-xr-x 1 root root   4096 Dec 10 17:17 .config
-rw-r--r-- 1 root root 249081 Dec 11 19:30 pic.jpg
drwxr-xr-x 1 root root   4096 Dec  2 22:04 sample_data
```

In [ ]:
```
!pip install pycuda
```

```
Collecting pycuda
  Downloading https://files.pythonhosted.org/packages/46/61/47d3235a4c13eec5a5f
03594ddb268f4858734e02980afbcd806e6242fa5/pycuda-2020.1.tar.gz (https://files.p
ythonhosted.org/packages/46/61/47d3235a4c13eec5a5f03594ddb268f4858734e02980afbc
d806e6242fa5/pycuda-2020.1.tar.gz) (1.6MB)
     |████████████████████████████████| 1.6MB 13.6MB/s
Collecting pytools>=2011.2
  Downloading https://files.pythonhosted.org/packages/b7/30/c9362a282ef89106768
cba9d884f4b2e4f5dc6881d0c19b478d2a710b82b/pytools-2020.4.3.tar.gz (https://file
s.pythonhosted.org/packages/b7/30/c9362a282ef89106768cba9d884f4b2e4f5dc6881d0c1
9b478d2a710b82b/pytools-2020.4.3.tar.gz) (62kB)
     |████████████████████████████████| 71kB 11.1MB/s
Requirement already satisfied: decorator>=3.2.0 in /usr/local/lib/python3.6/dis
t-packages (from pycuda) (4.4.2)
Collecting appdirs>=1.4.0
  Downloading https://files.pythonhosted.org/packages/3b/00/2344469e2084fb287c2
e0b57b72910309874c3245463acd6cf5e3db69324/appdirs-1.4.4-py2.py3-none-any.whl (h
ttps://files.pythonhosted.org/packages/3b/00/2344469e2084fb287c2e0b57b729103098
74c3245463acd6cf5e3db69324/appdirs-1.4.4-py2.py3-none-any.whl)
Collecting mako
  Downloading https://files.pythonhosted.org/packages/a6/37/0e706200d22172eb8fa
17d68a7ae22dec7631a0a92266634fb518a88a5b2/Mako-1.1.3-py2.py3-none-any.whl (http
s://files.pythonhosted.org/packages/a6/37/0e706200d22172eb8fa17d68a7ae22dec7631
a0a92266634fb518a88a5b2/Mako-1.1.3-py2.py3-none-any.whl) (75kB)
     |████████████████████████████████| 81kB 12.0MB/s
Requirement already satisfied: six>=1.8.0 in /usr/local/lib/python3.6/dist-pack
ages (from pytools>=2011.2->pycuda) (1.15.0)
Requirement already satisfied: numpy>=1.6.0 in /usr/local/lib/python3.6/dist-pa
ckages (from pytools>=2011.2->pycuda) (1.18.5)
Requirement already satisfied: dataclasses>=0.7 in /usr/local/lib/python3.6/dis
t-packages (from pytools>=2011.2->pycuda) (0.8)
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/python3.6/di
st-packages (from mako->pycuda) (1.1.1)
Building wheels for collected packages: pycuda, pytools
  Building wheel for pycuda (setup.py) ... done
  Created wheel for pycuda: filename=pycuda-2020.1-cp36-cp36m-linux_x86_64.whl
 size=620933 sha256=e89df869e8533e7e6c88b6b173ae422b3e55997ac2f020eed48ff80b881
d1992
  Stored in directory: /root/.cache/pip/wheels/8f/78/d1/5bb826f81d9d490297a348d
818ff3ee6dd6f2075b06dde6ea0
  Building wheel for pytools (setup.py) ... done
  Created wheel for pytools: filename=pytools-2020.4.3-py2.py3-none-any.whl siz
e=61374 sha256=1c08eb7bf18b96ead196687968767b52306c90f7add3e25a1a1231a9ac84269d
  Stored in directory: /root/.cache/pip/wheels/af/c7/81/a22edb90b0b09a880468b22
53bb1df8e9f503337ee15432c64
Successfully built pycuda pytools
Installing collected packages: appdirs, pytools, mako, pycuda
Successfully installed appdirs-1.4.4 mako-1.1.3 pycuda-2020.1 pytools-2020.4.3
```

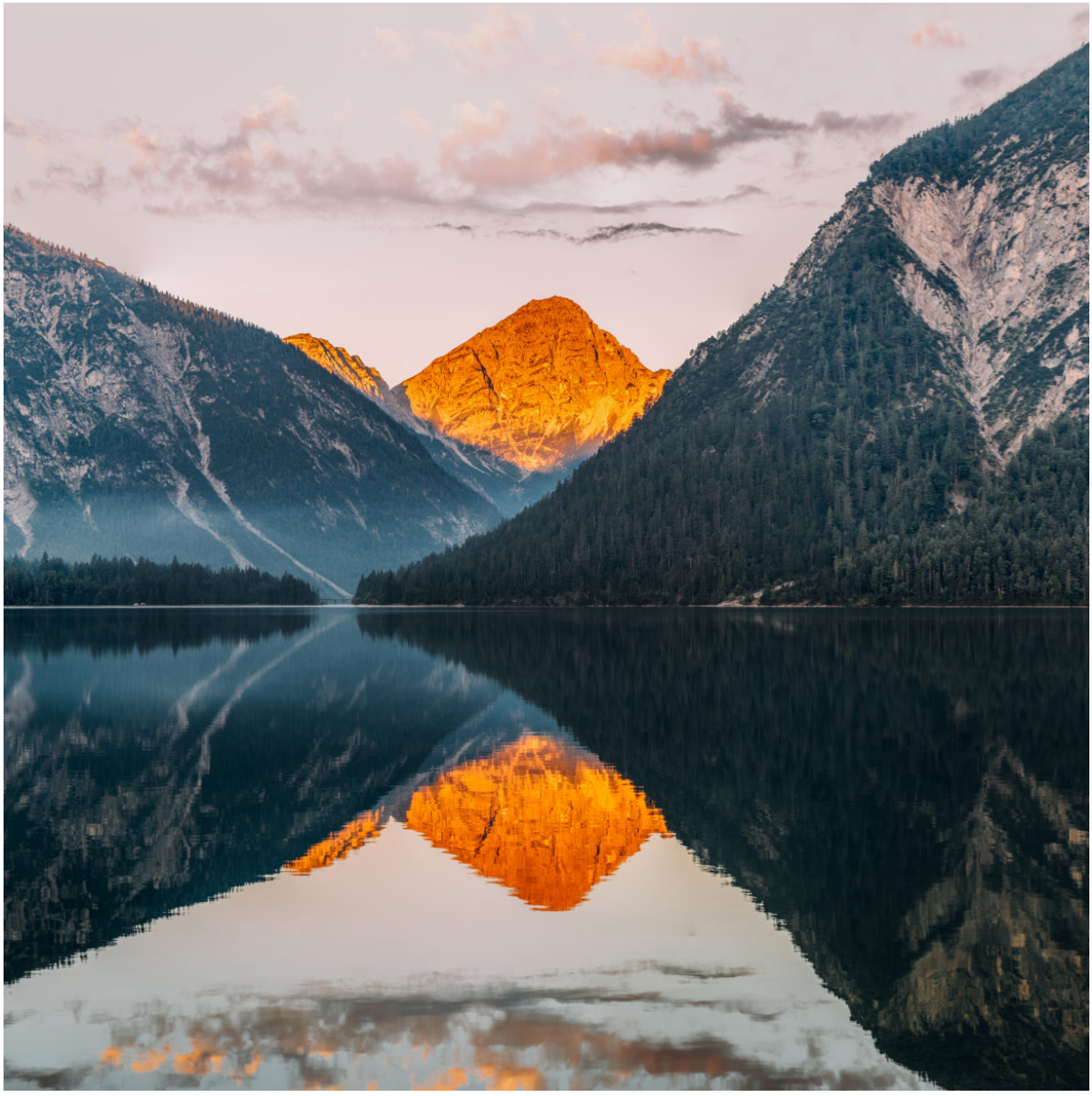
In [ ]:
```
!nvidia-smi
```

```
Fri Dec 11 19:31:49 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 455.45.01    Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   40C    P8    10W /  70W |      0MiB / 15079MiB |      0%      Default |
|                               |                      |                 ERR! |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Картинка

In [ ]:

```
import PIL.Image as img


image = img.open("pic.jpg")
display(image)
```



Размытие по шаблону

In [ ]:
```
TEMPLATE_FILTER = '''
__global__ void applyTemplateFilter(const unsigned char *inputChannel, unsigned c
                            const unsigned int height, const unsigned int width,
                            const float *filter, const unsigned int filterSize,
    const unsigned int row = threadIdx.x + blockIdx.x * blockDim.x;
    const unsigned int col = threadIdx.y + blockIdx.y * blockDim.y;
    if (row < height && col < width) {
        const int filterHalf = filterSize / 2;
        float blur = 0.0;
        for (int i = -filterHalf; i <= filterHalf; i++) {
            for (int j = -filterHalf; j <= filterHalf; j++) {
                const unsigned int x = max(0, min(height - 1, row + i));
                const unsigned int y = max(0, min(width - 1, col + j));

                const float w = filter[(i + filterHalf) * filterSize + (j + filte
                blur += w * inputChannel[x * width + y];
            }
        }
        outputChannel[row * width + col] = static_cast<unsigned char>(blur / filt
    }
}

'''
```

In [ ]:

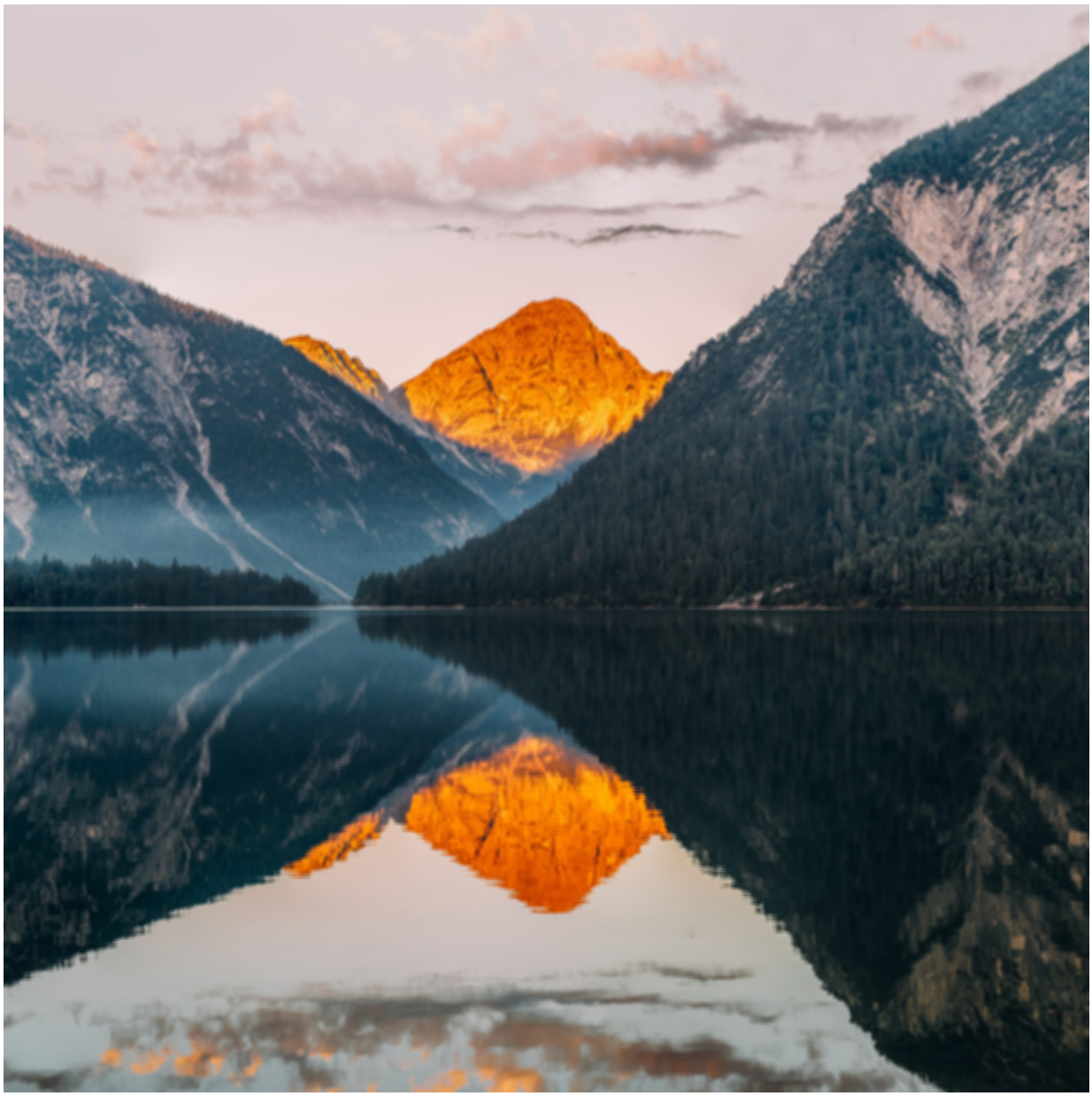
```
import math as mt

import numpy as np
import pycuda.autoinit
from pycuda import driver
from pycuda import compiler


BLOCK_SIZE = 16


def template_filter_cuda(src, filter):
    assert filter.shape[0] == filter.shape[1] and filter.shape[0] % 2
    filter_size = filter.shape[0]
    filter_sum = filter.sum()
    height, width = src.shape[:2]

    dim_grid_x = mt.ceil(height / BLOCK_SIZE)
    dim_grid_y = mt.ceil(width / BLOCK_SIZE)

    mod = compiler.SourceModule(TEMPLATE_FILTER)
    apply_template_filter = mod.get_function('applyTemplateFilter')

    res = np.empty_like(src)
    for color in range(src.shape[2]):
        channel = src[:, :, color].copy()
        apply_template_filter(
            driver.In(channel),
            driver.Out(channel),
            np.uint32(height),
            np.uint32(width),
            driver.In(filter),
            np.uint32(filter_size),
            np.float32(filter_sum),
            block=(BLOCK_SIZE, BLOCK_SIZE, 1),
            grid=(dim_grid_x, dim_grid_y)
        )
        res[:, :, color] = channel

    return res
```

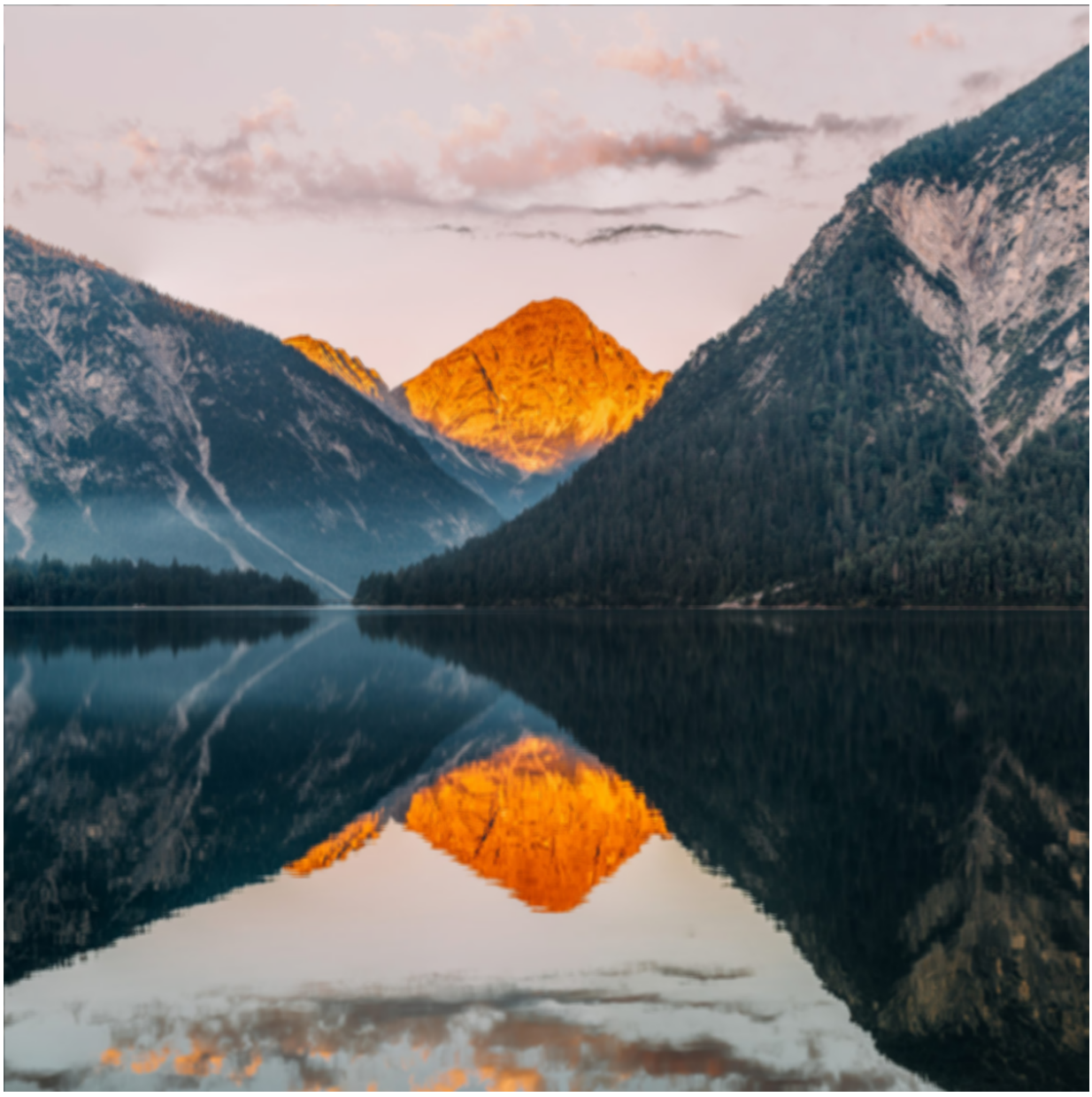
In [ ]:
```
def template_filter_cpu(src, filter):
    assert filter.shape[0] == filter.shape[1] and filter.shape[0] % 2
    filter_size = filter.shape[0]
    filter_sum = filter.sum()
    height, width = src.shape[:2]

    res = np.empty_like(src)

    filter_half = filter_size // 2
    for color in range(src.shape[2]):
      for row in range(height):
          for col in range(width):
                blur = 0.0
                for i in range(-filter_half, filter_half + 1):
                    for j in range(-filter_half, filter_half + 1):
                        x = max(0, min(height - 1, row + i))
                        y = max(0, min(width - 1, col + j))

                        w = filter[i + filter_half, j + filter_half]
                        blur += w * src[x, y, color];
                res[row, col, color] = int(blur / filter_sum)
    return res
```

In [ ]:
```
def perform_filter(image, cpu_filter_func, cuda_filter_func):
    src = np.array(image)

    def process_filter(src, filter_func):
        res = filter_func(src)
        res_image = img.fromarray(res)
        display(res_image)
        return res, res_image

    res_cpu, _ = process_filter(src, cpu_filter_func)
    res_cuda, _ = process_filter(src, cuda_filter_func)

    return res_cpu, res_cuda
```

In [ ]:

```
from functools import partial


filter = np.random.random((5, 5)).astype(np.float32)
image = img.open("pic.jpg")

res_cpu, res_cuda = perform_filter(
    image,
    partial(template_filter_cpu, filter=filter),
    partial(template_filter_cuda, filter=filter),
)
```





Медианный фильтр

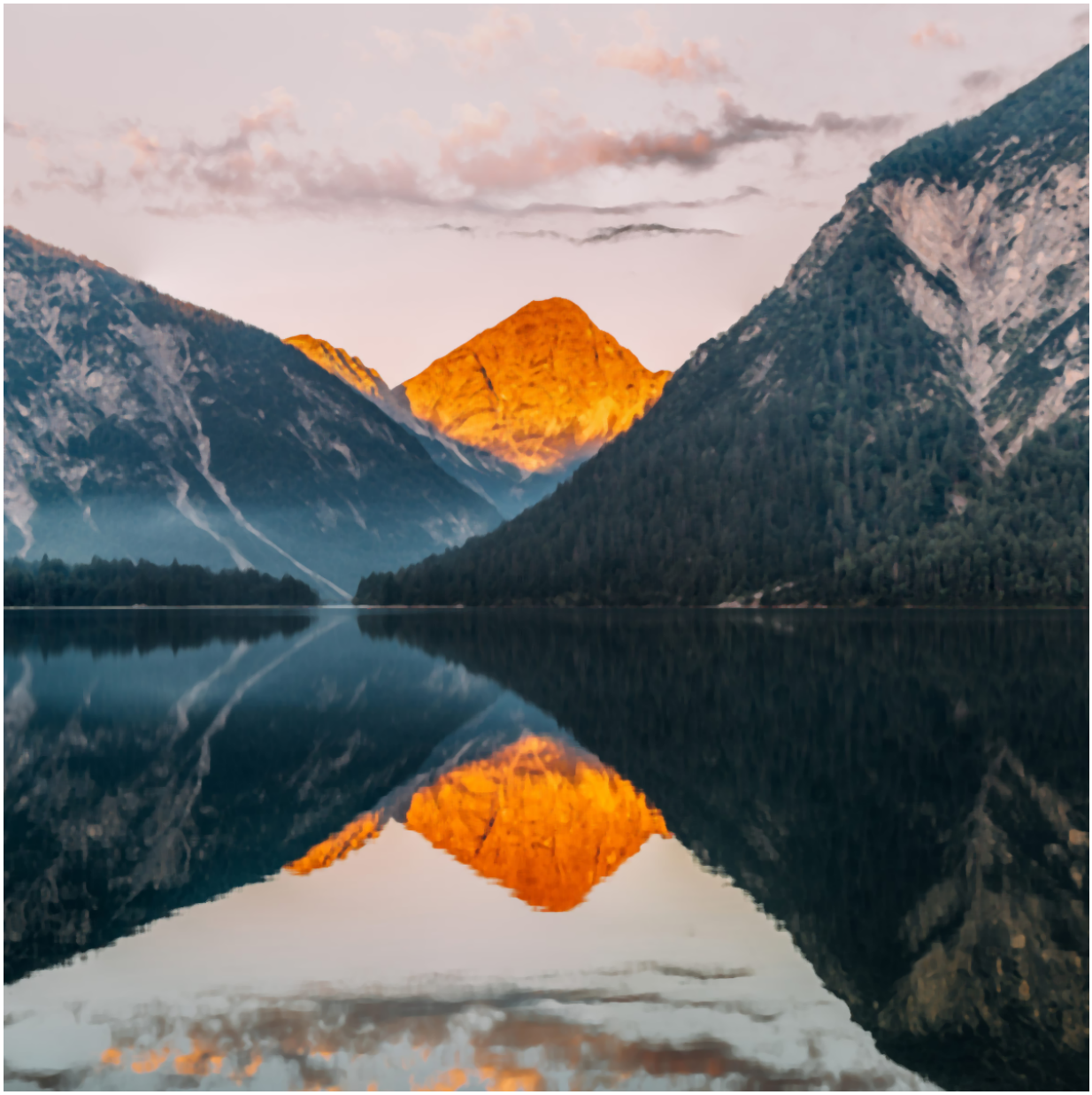
In [ ]:

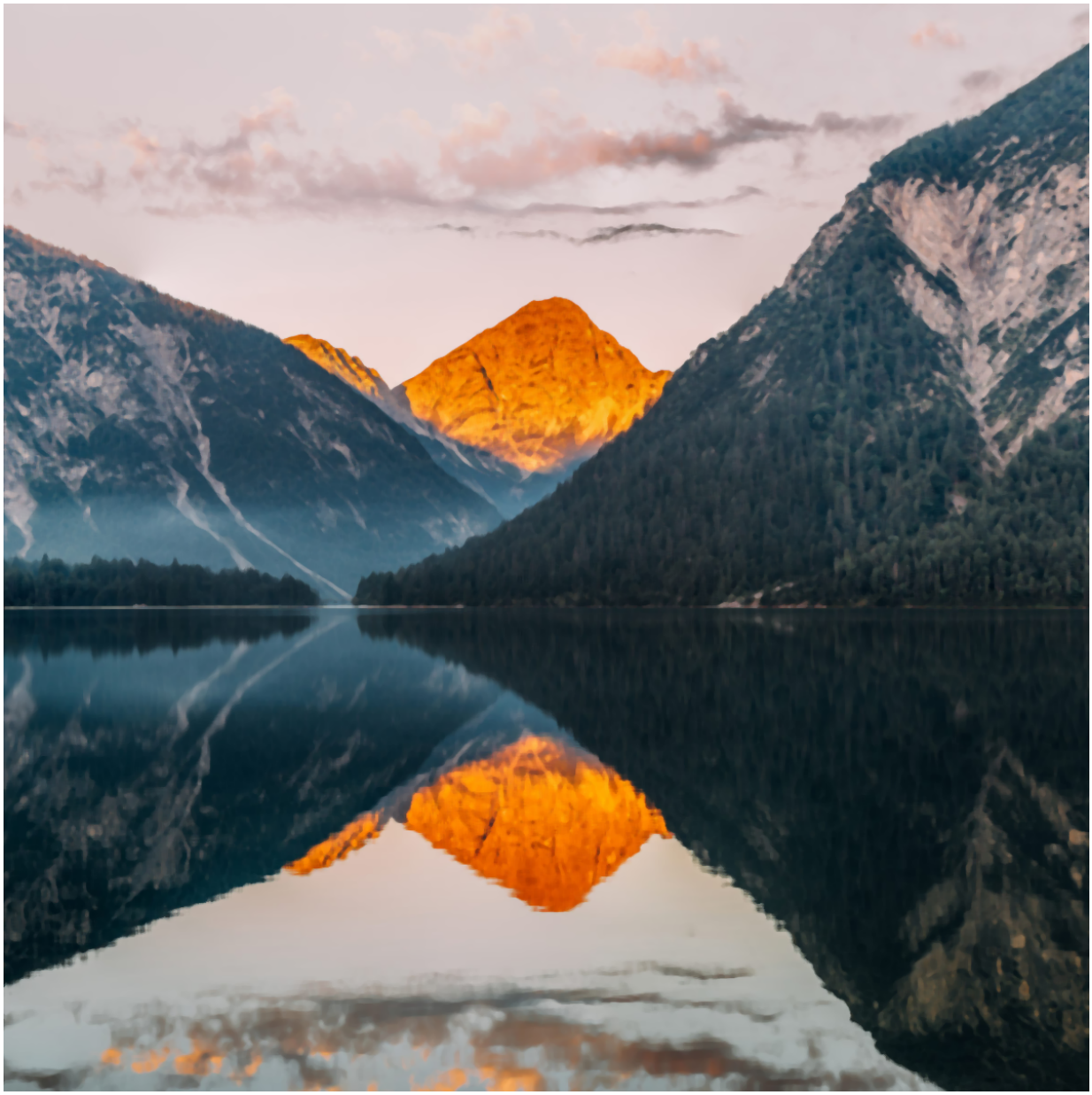
```
MEDIAN_FILTER = '''
__global__ void applyMedianFilter(const unsigned char *inputChannel, unsigned cha
                                  const unsigned int height, const unsigned int width,
                                  const unsigned int filterSize) {
    const unsigned int row = threadIdx.x + blockIdx.x * blockDim.x;
    const unsigned int col = threadIdx.y + blockIdx.y * blockDim.y;
    if (row < height && col < width) {
        const int filterHalf = filterSize / 2;
        unsigned char* values = new unsigned char[filterSize * filterSize];
        for (int i = -filterHalf; i <= filterHalf; i++) {
            for (int j = -filterHalf; j <= filterHalf; j++) {
                const unsigned int x = max(0, min(height - 1, row + i));
                const unsigned int y = max(0, min(width - 1, col + j));

                values[(i + filterHalf) * filterSize + (j + filterHalf)] = inputC
            }
        }
        for (unsigned int i = 0; i <= filterSize * filterSize / 2; i++) {
            unsigned int pos = i;
            for (unsigned j = i + 1; j < filterSize * filterSize; j++) {
                if (values[pos] > values[j]) {
                    pos = j;
                }
            }
            unsigned char tmp = values[pos];
            values[pos] = values[i];
            values[i] = tmp;
        }
        outputChannel[row * width + col] = values[filterSize * filterSize / 2];
        delete[] values;
    }
}

'''
```

In [ ]:
```python
def median_filter_cuda(src, filter_size):
    assert filter_size % 2
    height, width = src.shape[:2]

    dim_grid_x = mt.ceil(height / BLOCK_SIZE)
    dim_grid_y = mt.ceil(width / BLOCK_SIZE)

    mod = compiler.SourceModule(MEDIAN_FILTER)
    apply_template_filter = mod.get_function('applyMedianFilter')

    res = np.empty_like(src)
    for color in range(src.shape[2]):
        channel = src[:, :, color].copy()
        apply_template_filter(
            driver.In(channel),
            driver.Out(channel),
            np.uint32(height),
            np.uint32(width),
            np.uint32(filter_size),
            block=(BLOCK_SIZE, BLOCK_SIZE, 1),
            grid=(dim_grid_x, dim_grid_y)
        )
        res[:, :, color] = channel

    return res
```

In [ ]:
```python
def median_filter_cpu(src, filter_size):
    assert filter_size % 2
    height, width = src.shape[:2]

    res = np.empty_like(src)

    filter_half = filter_size // 2
    for color in range(src.shape[2]):
      for row in range(height):
          for col in range(width):
                lst = []
                for i in range(-filter_half, filter_half + 1):
                    for j in range(-filter_half, filter_half + 1):
                        x = max(0, min(height - 1, row + i))
                        y = max(0, min(width - 1, col + j))

                        lst.append(src[x, y, color])
                res[row, col, color] = sorted(lst)[filter_size ** 2 // 2 + 1]
    return res
```

In [ ]:
```
image = img.open("pic.jpg")

res_cpu, res_cuda = perform_filter(
    image,
    partial(median_filter_cpu, filter_size=5),
    partial(median_filter_cuda, filter_size=5),
)
```





In [ ]: