

Инженерная школа информационных технологий и робототехники (ИШИТР)

Направление подготовки Искусственный интеллект и машинное обучение

Отделение школы (НОЦ) 09.04.01 Информатика и вычислительная техника

**Лабораторная работа**

По дисциплине «Параллельные и высокопроизводительные вычисления»

|  |
| --- |
| **Тема работы** |
| Использование многопоточных вычислений при программировании видеопроцессоров |

Студент

|  |  |  |  |
| --- | --- | --- | --- |
| **Группа** | **ФИО** | **Подпись** | **Дата** |
| 8ВМ13 | Е.М. Якунин |  | 05.05.2022 |

Принял

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Должность** | **ФИО** | **Ученая степень, звание** | **Подпись** | **Дата** |
| К.Т.Н. Доцент (ОИТ, ИШИТР) | С.В. Аксёнов |  |  |  |

Томск – 2022 г.

# Задание

1. Корректировка цветовых каналов (вариант 3).

Получить модифицированный синий и красный каналы на изображении согласно следующим выражениям для синего и красного цветовых компонент соответственно:

,

где – интенсивность пикселя *v*, – значение красной компоненты пикселя , – значение зелёной компоненты пикселя , – значение синей компоненты пикселя .

# Отчет

Лабораторная работа была реализована в google.colab с использованием языка программирования «Python». Для выполнения данного задания были применены библиотеки *numpy*, *time* и *pytorch*.

Сначала импортируем необходимые библиотеки и загружаем изображения:

import torch

import numpy as np

import matplotlib.pyplot as plt

import time, threading

from threading import Thread

from PIL import Image

from google.colab import files

import math

print(torch.cuda.device\_count())

print(torch.cuda.get\_device\_name(0))

1

Tesla K80

uploaded = files.upload()

 **1024х768(3).jpg**(image/jpeg) - 207802 bytes, last modified: 29.04.2022 - 100% done

 **1280x960н.jpg**(image/jpeg) - 252987 bytes, last modified: 05.05.2022 - 100% done

 **2048x1536.jpg**(image/jpeg) - 1322201 bytes, last modified: 05.05.2022 - 100% done

Saving 1024х768(3).jpg to 1024х768(3).jpg

Saving 1280x960н.jpg to 1280x960н.jpg

Saving 2048x1536.jpg to 2048x1536.jpg

Инициализируем переменные для изображений и отобразим их (рис.1):

# Выводим исходное изображение

images = []

for i in uploaded.keys():

  images.append(i)

print(images)

my\_image = np.array(Image.open(images[0]))

image\_1280x960 = np.array(Image.open(images[1]))

image\_2048x1536 = np.array(Image.open(images[2]))

print('1024x768: \n')

plt.imshow(my\_image)

plt.colorbar()

plt.show()

print('1280x960: \n')

plt.imshow(image\_1280x960)

plt.colorbar()

plt.show()

print('2048x1536: \n')

plt.imshow(image\_2048x1536)

plt.colorbar()

plt.show()

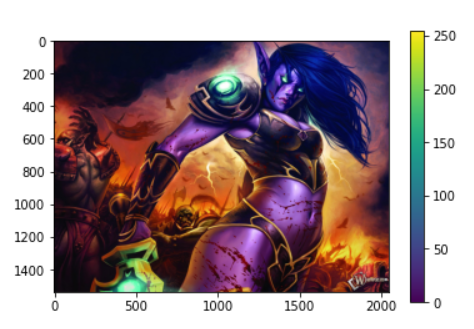
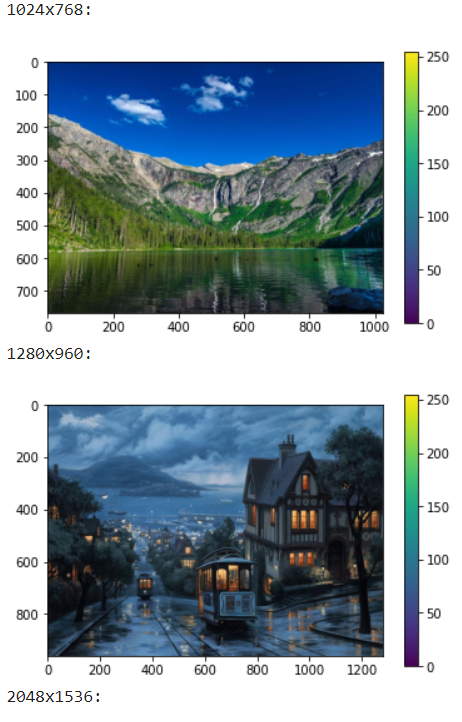


Рисунок 1 ‒ Исходные изображения

Создадим 3 тензора pytorch для каждого изображения и передадим данные сразу на видеокарту:

cuda0 = torch.device('cuda:0')

torch\_img = torch.tensor(img, device=cuda0)

torch\_img\_1280x960 = torch.tensor(image\_1280x960, device=cuda0)

torch\_img\_2048x1536 = torch.tensor(image\_2048x1536, device=cuda0)

Получим новый цветовой канал, выведем изображение и замерим потраченное время на вычисления gpu, при троекратном перезапуске:

torch\_img\_newblue = torch.zeros\_like(torch\_img, device=cuda0)

start = time.time()

for i in range(3):

  torch\_img\_newblue[:,:,2] = torch\_img[:,:,2] - ((torch\_img[:,:,1] + torch\_img[:,:,2]) / 2)

  torch\_new\_img = torch\_img[:,:,0] + torch\_img[:,:,1] + torch\_img\_newblue[:,:,2]

print(f"Затраченное время: {time.time()-start}. \n")

plt.imshow(torch\_new\_img.cpu())

plt.colorbar()

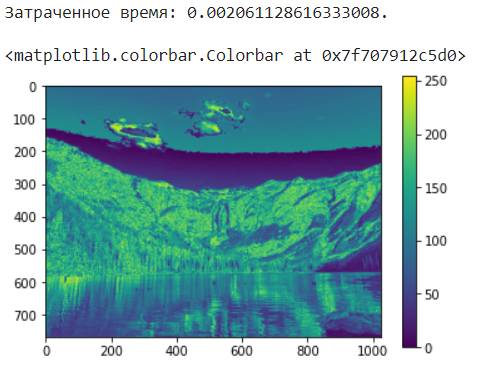


Рисунок 2 ‒ Новый синий цветовой канал для изображения 1024х768

torch\_img\_newred = torch.zeros\_like(torch\_img, device=cuda0)

start = time.time()

for i in range(3):

  torch\_img\_newred[:,:,0] = torch\_img[:,:,0] + torch\_img[:,:,1] - 2 \* (abs(torch\_img[:,:,0] - torch\_img[:,:,1]) + torch\_img[:,:,2])

  torch\_new\_img = torch\_img\_newred[:,:,0] + torch\_img[:,:,1] + torch\_img[:,:,2]

print(f"Затраченное время: {time.time()-start}. \n")

plt.imshow(torch\_new\_img.cpu())

plt.colorbar()

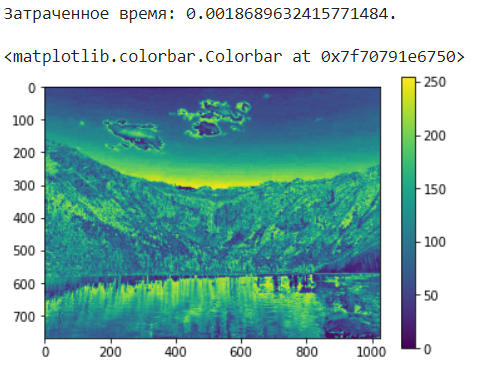


Рисунок 3 ‒ Новый красный цветовой канал для изображения 1024х768

Повторим действия для оставшихся изображений. Создадим таблицу с замерами потраченного времени для каждого изображения.

Таблица 1. Затраченное время gpu на выполнение задачи

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Изображение | Синий цвет | | | Красный цвет | | |
| 1 замер | 2 замер | 3 замер | 1 замер | 2 замер | 3 замер |
| 1024х768 | 0,0020 | 0,0014 | 0,0007 | 0.0020 | 0.0013 | 0,0018 |
| 1280х960 | 0,0010 | 0,0021 | 0,0013 | 0,0021 | 0,0026 | 0,0032 |
| 2048х1536 | 0,0013 | 0,0012 | 0,0018 | 0,0010 | 0,0035 | 0,0024 |

# Результаты работы

С помощью библиотеки PyTorch, мы написали небольшой и простой код, который позволяет нам вычислять цветовой канал и строить новое изображение с ним, при помощи вычислительных мощностей видеопроцессоров. По полученным данным (табл.1), видим, что это заняло у нас считанные микросекунды, что очень быстро. Вычисление на gpu может сэкономить нам очень много времени при работе с матрицами, когда данные во много раз превышают наши и требуют часы работы на cpu.

# Выводы

В результате выполненной работы была написана небольшая программа для нахождения новых цветовых канало и построения с ними изображения. Процесс вычисления выполнялся с использованием gpu. Оценивание работы происходило замером времени выполнения расчетов при троекратном запуске. Были получены базовые знания работы с библиотекой «PyTorch», для распараллеливания процессов на языке Python. Познакомились с темой многопоточности на GPU, а также с технологией «CUDA».

По полученным итогам был сделан вывод, что использование gpu для выполнения несложных операций имеет практическую выгоду. GPU может помочь ускорить процессы, при выполнении больших по количеству и несложных операций.