МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра 307 «Цифровые технологии и информационные системы»

Домашнее задание №2 по курсу «Мультимедиа технологии»

Тема 29: «Многомасштабные преобразования. Пирамида изображений»

Задание подготовил студент группы МЗО-412Б-19:

Журавков Владислав

Максимович

_____

дата, подпись

Задание принял:

Минасян Виталий Борисович

_____

дата, подпись

Москва 2023

Оглавление

Теоретическая часть

Пирамида изображений — класс кратномасштабных иерархических структур данных, разработанных для применения в прикладных задачах машинного зрения, сжатия информации, анализа текстур растровых изображений и т. п. Такие структуры содержат на каждом своём уровне иерархии уменьшенную версию предыдущего изображения, каждая из которых рекурсивно вычисляется на базе предыдущих слоёв посредством применения однотипной операции (например — сглаживания). При этом всякому уровню иерархии ставится в соответствие актуальный для решаемой задачи параметр масштаба, который описывает интересующие свойства изображения.

В рамках задания будет рассмотрено 2 типа пирамид изображения:

1. Гауссова пирамида изображения
2. Пирамида изображения Лапласа

Гауссова пирамида изображения - это популярная техника, используемая в компьютерном зрении и обработке изображений для создания многомасштабного представления изображения. Она основана на концепции гауссовского размытия, которое представляет собой операцию сглаживания, уменьшающую шум в изображении.

Чтобы создать пирамиду гауссовых изображений, исходное изображение сначала свертывается с гауссовым фильтром для получения сглаженной версии изображения. Затем сглаженное изображение уменьшается, или изменяется до меньшего размера, чтобы создать новое изображение с меньшим разрешением. Этот процесс может быть повторен для создания серии изображений, каждое из которых имеет более низкое разрешение, чем предыдущее. Полученный набор изображений называется пирамидой изображений, так как изображения располагаются в пирамидоподобной структуре.

Гауссова пирамида изображений имеет несколько полезных применений в обработке изображений и компьютерном зрении. Она может использоваться для регистрации изображений, обнаружения объектов и сегментации изображений. Используя многомасштабное представление изображения, можно обнаруживать объекты разного масштаба и сегментировать изображение на области разного размера.

В целом, гауссова пирамида изображений является мощным инструментом для обработки изображений и извлечения из них полезной информации. Ее способность создавать многомасштабное представление изображения делает ее ценным методом во многих приложениях компьютерного зрения и обработки изображений.

Пирамида изображений Лапласа - это еще одна техника, широко используемая в компьютерном зрении и обработке изображений, которая тесно связана с пирамидой изображений Гаусса. В ее основе лежит идея разложения изображения на ряд частотных полос, каждая из которых представляет собой определенный уровень детализации.

Чтобы создать пирамиду изображений Лапласа, мы начинаем с построения гауссовой пирамиды, как описано ранее, которая состоит из набора изображений разного масштаба. Затем мы используем эту пирамиду для построения пирамиды Лапласа путем вычитания каждого уровня пирамиды Гаусса из следующего уровня, дискретизированного до того же размера. Это создает серию разностных изображений, которые представляют высокочастотные компоненты изображения.

Результирующая пирамида Лапласа имеет такое же количество уровней, как и пирамида Гаусса, но каждый уровень содержит информацию о высокочастотных деталях, которые были удалены в пирамиде Гаусса. Самый нижний уровень пирамиды Лапласа содержит ту же информацию, что и самый нижний уровень пирамиды Гаусса, но более высокие уровни содержат информацию о краях и мелких деталях изображения.

Пирамида Лапласа может использоваться для решения различных задач обработки изображений, таких как сжатие, слияние и улучшение изображений. Она особенно полезна при сжатии изображений, поскольку высокочастотные компоненты изображения могут быть отброшены или квантованы более активно, чем низкочастотные компоненты, что приводит к более эффективному сжатию.

В целом, пирамида изображений Лапласа - это мощный метод обработки изображений и компьютерного зрения, который можно использовать в сочетании с другими методами для извлечения полезной информации из изображений и выполнения различных задач.

Обсуждение результатов

Разработано программное обеспечение (приложение с пользовательским интерфейсом), имеющее функционал построения пирамид изображения

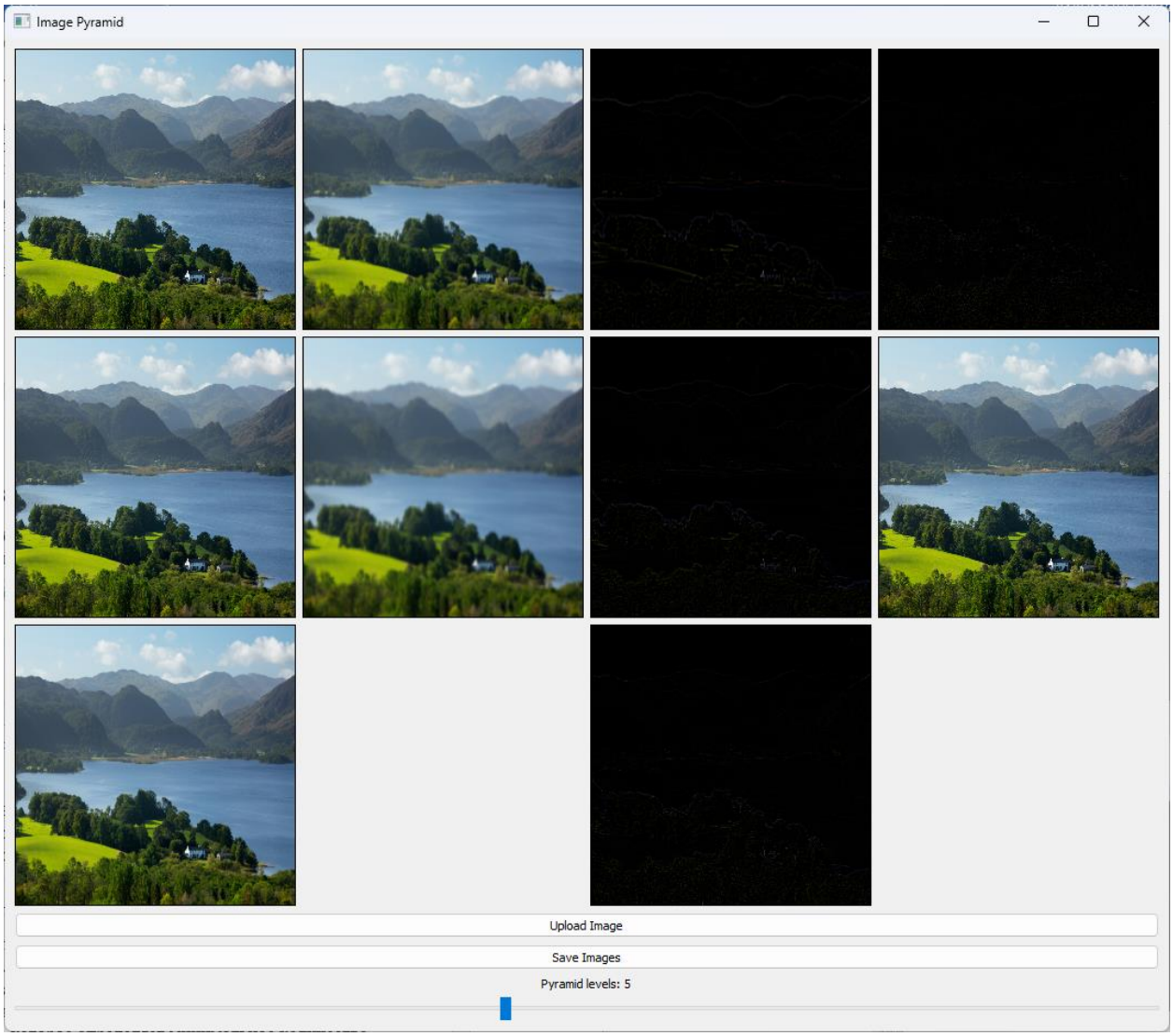

Рисунок 1 Окно программы

Приложение разделено на 3 секции:

1. Область построения пирамиды Гаусса
2. Область построения пирамиды Лапласа
3. Панель пользовательского взаимодействия

На панели пользовательского взаимодействия расположены кнопки:

1. Upload Image – для выбора изображения для построения пирамид
2. Save Image – для сохранения изображений, получившихся в результате алгоритма (изображения будут сохранены в оригинальных масштабах)
3. Pyramid levels – ползунок для регулировки уровней пирамид.

При попытке выбрать файл для построения пирамид видим предупреждение:

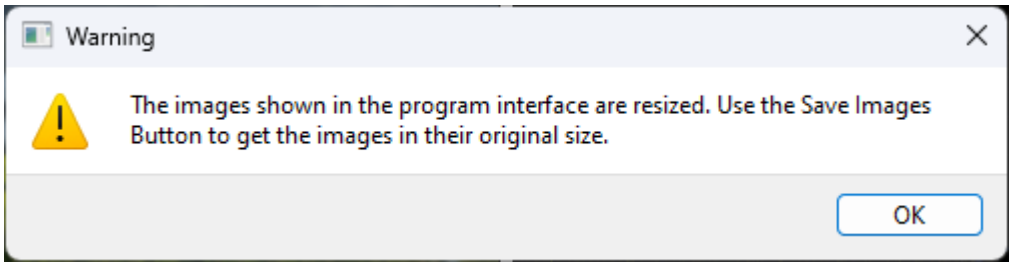

Рисунок 2 Предупреждение о масштабировании изображений в окне программы

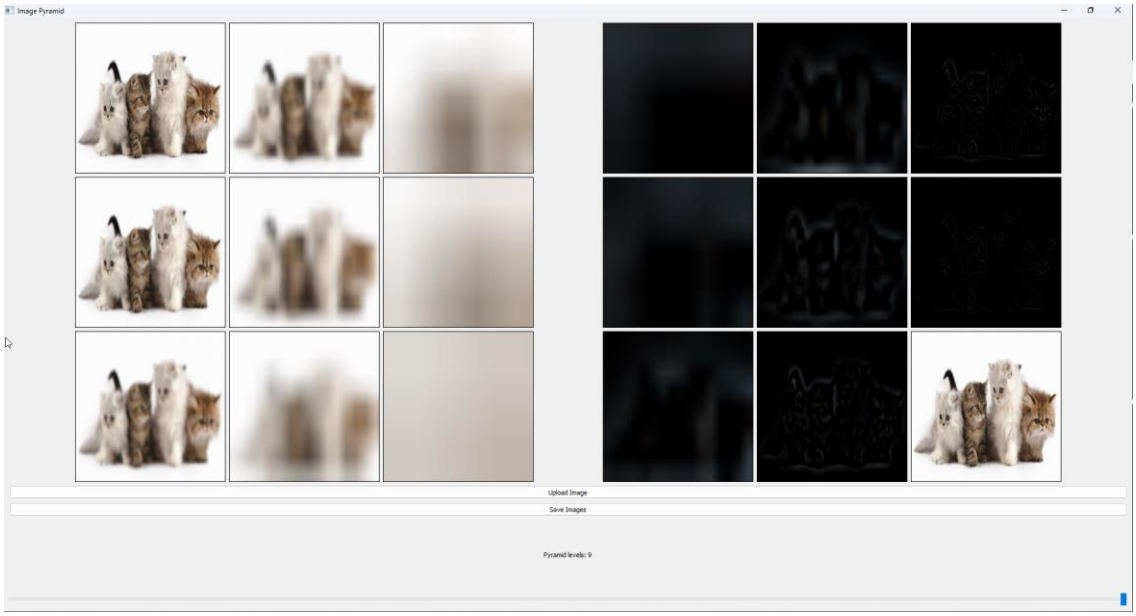

Рисунок 3 Построенные пирамиды изображений, приведённые к одному масштабу Котики

Рисунок 4 Построенные пирамиды изображений, приведённые к одному масштабу. Пейзаж

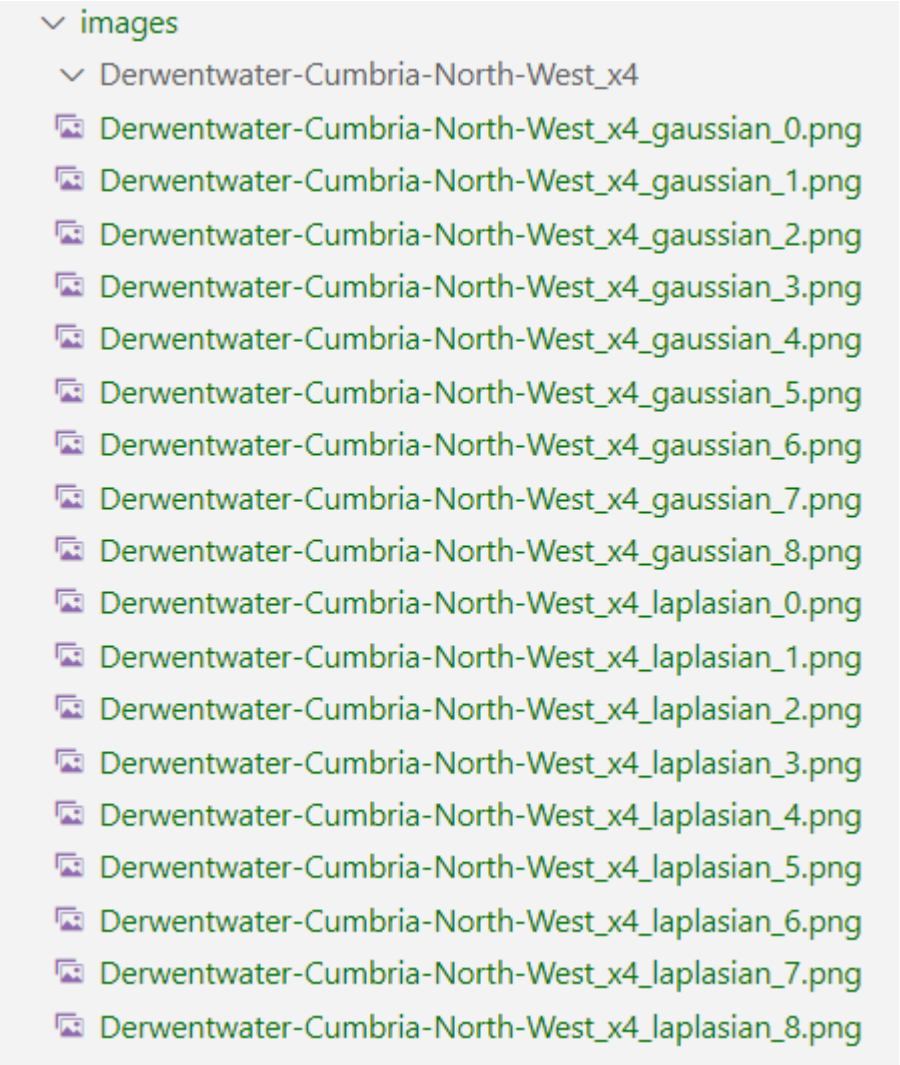

Рисунок 4 Список полученных изображений после сохранения

Максимальный размер изображения ограничивается размером буфера вычислительной машины. Для используемой конфигурации этот лимит: 30.000 x 30.000 пикселей (изображение 2.5 гб).
Поддерживаемые форматы изображений: .bmp .png .tif .jpg .hdr
Максимальная глубина цвета: 24 бит на канал.

Конфигурация:
Процессор: 3,3 GHz 8-ядерный процессор Intel Core i7-1360H
Графика: GeForce RTX 3050 Laptop GPU GDDR6 4GB
Память: 16 ГБ 3200 MHz SODIMM
Накопитель: NVMe SSD накопитель SAMSUNG ёмкостью 500 ГБ

Ссылка на репозиторий: https://github.com/Shuich1/media-technology/tree/main/image_pyramid

Для корректной работы программы необходим установленный интерпретатор Python, а также понадобится установить дополнительные зависимости из файла requirements.txt:

```
pip install -r requirements.txt
```

Запуск программы осуществляется командой:

```
python main.py
```

Листинг программы

```
import os
import sys

import cv2
import numpy as np
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QImage, QPixmap
from PyQt5.QtWidgets import (QApplication, QFileDialog, QGridLayout,
                             QHBoxLayout, QLabel, QMainWindow, QMessageBox,
                             QPushButton, QSlider, QVBoxLayout, QWidget)


class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Image Pyramid")

        self.move(self.frameGeometry().width() // 2,
                  self.frameGeometry().height() // 2)

        self.image_button = QPushButton("Upload Image")
        self.image_button.clicked.connect(self.upload_image)
        self.save_images_button = QPushButton("Save Images")
        self.save_images_button.clicked.connect(self.save_images)
        self.save_images_button.setEnabled(False)

        self.slider = QSlider(Qt.Horizontal)
        self.slider.setMinimum(2)
        self.slider.setMaximum(9)
        self.slider.setTickInterval(1)
        self.slider.setValue(3)
        self.slider.valueChanged.connect(self.update_slider)

        self.slider_label = QLabel()
        self.slider_label.setAlignment(Qt.AlignCenter)
        self.slider_label.setText(f"Pyramid levels: {self.slider.value()}")

        self.central_widget = QWidget()
        self.main_layout = QVBoxLayout()
        self.layout = QHBoxLayout()
        self.left_layout = QGridLayout()
        self.right_layout = QGridLayout()

        self.layout.addLayout(self.left_layout)
        self.layout.addLayout(self.right_layout)

        self.main_layout.addLayout(self.layout)
        self.main_layout.addWidget(self.image_button)
        self.main_layout.addWidget(self.save_images_button)
        self.main_layout.addWidget(self.slider_label)
```

```
        self.main_layout.addWidget(self.slider)

        self.central_widget.setLayout(self.main_layout)
        self.setCentralWidget(self.central_widget)

        self.image_path = None
        self.gaussian_pyramid_images = []
        self.laplacian_pyramid_images = []

    def update_slider(self):
        self.slider_label.setText(f"Pyramid levels: {self.slider.value()}")
        self.update_images()

    def upload_image(self):
        self.image_path, _ = QFileDialog.getOpenFileName(
            self, "Open Image", "", "Image Files (*.png *.jpg)")
        if self.image_path:
            QMessageBox.warning(
                self,
                "Warning",
                "The images shown in the program interface are resized. Use the
Save Images Button to get the images in their original size."
            )
            self.save_images_button.setEnabled(True)
        else:
            self.save_images_button.setEnabled(False)

        self.update_images()

    def update_images(self):
        self.clear_layouts()
        self.gaussian_pyramid_images = []
        self.laplacian_pyramid_images = []

        if self.image_path:
            try:
                image = cv2.imread(self.image_path)
            except Exception as e:
                QMessageBox.critical(self, "Error", "Can`t process this image,
maybe it`s too big or have incorrect file extension")
                return

            self.gaussian_pyramid_images = self.gaussian_pyramid(image)
            for idx, img in enumerate(self.gaussian_pyramid_images):
                label = QLabel()
                label.setFixedSize(256, 256)
                label.setAlignment(Qt.AlignCenter)
                label.setStyleSheet("border: 1px solid black;")

                img = cv2.resize(img, (label.width(), label.height()))
```

```
                qimage = QImage(img.data, img.shape[1], img.shape[0],
QImage.Format_RGB888).rgbSwapped()
                pixmap = QPixmap.fromImage(qimage)


                label.setPixmap(pixmap)

                self.left_layout.addWidget(label, idx % 3, idx // 3)

            self.laplacian_pyramid_images =
self.laplacian_pyramid(self.gaussian_pyramid_images)
            for idx, img in enumerate(self.laplacian_pyramid_images):
                label = QLabel()
                label.setFixedSize(256, 256)
                label.setAlignment(Qt.AlignCenter)
                label.setStyleSheet("border: 1px solid black;")

                img = cv2.resize(img, (label.width(), label.height()))

                qimage = QImage(img.data, img.shape[1], img.shape[0],
QImage.Format_RGB888).rgbSwapped()
                pixmap = QPixmap.fromImage(qimage)


                label.setPixmap(pixmap)

                self.right_layout.addWidget(label, idx % 3, idx // 3)

    def save_images(self):
        image_name = self.image_path.split('/')[-1].split(".")[0]

        directory_path =
f'{os.path.dirname(os.path.realpath(__file__))}\images\{image_name}'

        if not os.path.exists(directory_path):
            os.mkdir(directory_path)

        for idx, img in enumerate(self.gaussian_pyramid_images):
            save_path = os.path.join(
                directory_path, f"{image_name}_gaussian_{idx}.png"
            )
            cv2.imwrite(save_path, img)

        for idx, img in enumerate(self.laplacian_pyramid_images):
            save_path = os.path.join(
                directory_path, f"{image_name}_laplasian_{idx}.png"
            )
            cv2.imwrite(save_path, img)
```

```
    def gaussian_pyramid(self, image):
        tmp_img = image.copy()
        pyramid = []
        for i in range(self.slider.value()):
            new_img = cv2.pyrDown(tmp_img)
            pyramid.append(new_img)
            tmp_img = new_img.copy()

        return pyramid

    def laplacian_pyramid(self, gaussian_pyramid):
        pyramid = []
        for i in range(len(gaussian_pyramid) - 1, 0, -1):
            gaussian_expanded = cv2.pyrUp(gaussian_pyramid[i])
            height, width = gaussian_pyramid[i - 1].shape[:2]
            laplacian = cv2.subtract(gaussian_pyramid[i - 1],
cv2.resize(gaussian_expanded, (width, height)))
            pyramid.append(laplacian)

        pyramid.append(gaussian_pyramid[0])
        return pyramid

    def clear_layouts(self):
        for i in reversed(range(self.left_layout.count())):
            self.left_layout.itemAt(i).widget().setParent(None)

        for i in reversed(range(self.right_layout.count())):
            self.right_layout.itemAt(i).widget().setParent(None)


if __name__ == "__main__":
    app = QApplication(sys.argv)
    main_window = MainWindow()
    main_window.show()
    sys.exit(app.exec_())
```