Обработка видеопотоков с заменой фона с использованием фреймворка MediaPipe

Курсовой проект

Бинцаровский Леонид Петрович

Белорусский государственный университет
ФПМИ, ДМА, 3 курс
руководитель: старший преподаватель Пиршту к Д. И.

Минск, 2023

Применение замены фона на видеопотоке

- Рекламные материалы
- Графический дизайн
- Игры и симуляторы
- Увлекательные видеоролики
- Видеозвонки

Методы сегментации изображения и виды предобученных моделей

1. Выделение краёв;
2. Методы разреза графа;
3. Сегментация с помощью предобученной модели:
   - 3.1 **Semantic segmentation**;
   - 3.2 **Hair segmentation**;
   - 3.3 **Multi-class segmentation**;
   - 3.4 **Instance segmentation**.

Постановка задачи

Для реализации задачи обработки видеопотока с заменой фона, будут использоваться:

Модель для сегментации

Semantic segmentation

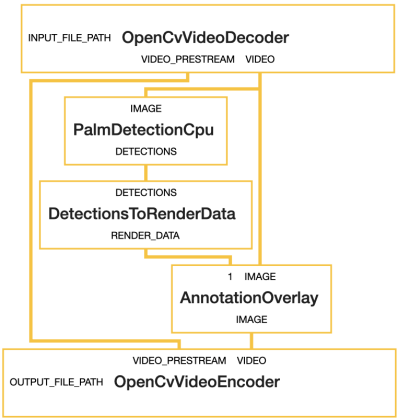
Фреймворк

MediaPipe

Язык программирования, среда разработки и операционная система

C++, Visual Studio и Windows

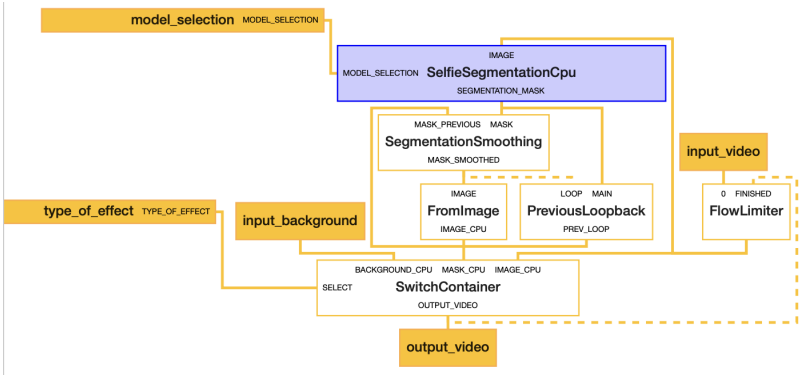
Основные элементы конвейера



1. Пакет (Packet)
2. Узлы (Nodes or calculator)
3. Потоки (Streams)
4. Подграф (Subgraph)
5. Граф (Graph)

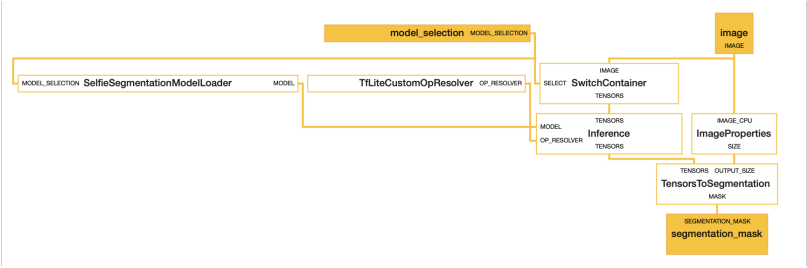
Реализация архитектуры конвейера

В ходе разработки конвейера был реализован граф Selfie_segmentation_cpu_ultimate.pbtxt:



Реализация архитектуры конвейера

А также подграф Selfie_segmentation_cpu.pbtxt:



Реализация калькуляторов

Калькулятор представляет собой класс C++, реализующий интерфейс CalculatorBase. В данном классе обязательно присутствуют следующие функции:

1. **::mediapipe::Status GetContract(CalculatorContract* cc)**;
2. **::mediapipe::Status Process(CalculatorContext* cc)**;
3. **::mediapipe::Status Open(CalculatorContext* cc)**;

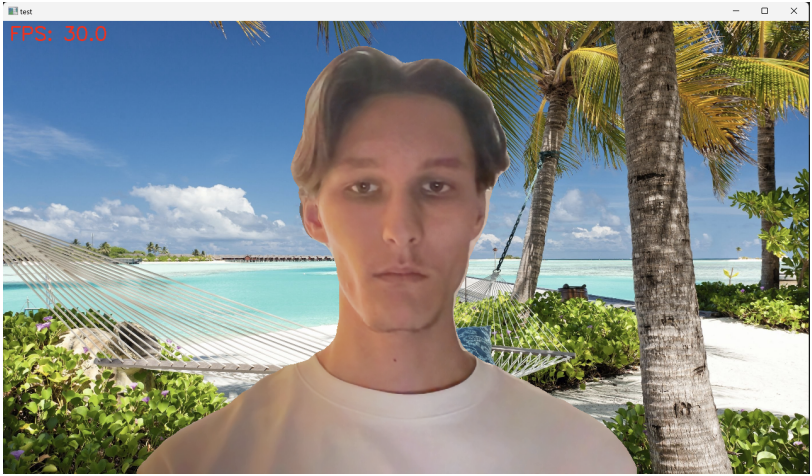
Помимо самого кода калькулятора необходимо определить proto-файл с конфигурацией калькулятора.

Реализация классов для работы в среде Visual Studio

```
class GMOD : public IGMOD {
public:
  virtual bool getOverlay() override;
  virtual void setOverlay(bool x) override;

  virtual void setCameraProps(const int& camId,
                              const int& camResX,
                              const int& camResY,
                              const int& camFps) override;

  virtual void start(cv::Mat& camera_frame_raw, bool& _load_flag) override;
  virtual void init(const std::string& filename,
                    cv::VideoCapture& capture,
                    const std::string& path = "background.jpg",
                    const int& typeOfEffect = 0,
                    const int& typeOfModel = 1) override;

private:
  absl::Status _runMPPGraph(cv::Mat& camera_frame_raw, bool& _load_flag);
  absl::Status _initMPPGraph(cv::VideoCapture& capture);

private:
  int camId;
  int camFps;
  int camResX;
  int camResY;
  int typeOfModel;
  int typeOfEffect;
  bool showOverlay;
  cv::Mat background;
  std::string graphFilename;
  std::string backgroundFilename;

  std::unique_ptr<mediapipe::OutputStreamPoller> output_poller;

  const char kInputStream[12] = "input_video";
  const char kOutputStream[13] = "output_video";
  const char kInputBackground[17] = "input_background";

public:
  std::shared_ptr<mediapipe::CalculatorGraph> _graph;
};
```

В большинстве своем в фреймворке MediaPipe используются макросы, которые возвращают ошибки типа absl::Status из фреймворка MediaPipe и их нельзя обработать напрямую в коде программы в Visual Studio. Поэтому были реализованы вспомогательные функции в private, выполняющие основной функционал, и основные функции (будут вызываться в основной программе), которые являются обертками функций из private и обрабатывают возможные ошибки.

Результат работы



Выводы

- Была рассмотрена задача обработки видеопотока с помощью фреймворка MediaPipe, а также ее применение на прикладном уровне;
- Сделан краткий обзор фреймворка MediaPipe;
- Выполнен обзор основных элементов конвейера фреймворка MediaPipe;
- Резработана архитектура ковейера и реализованы необходимые калькуляторы для него;
- Реализованы классы для работы с фреймфорком MediaPipe в среде разработки Visual Studio;
- Успешно протестированы результаты работы.