ООО «ТомскСофт-Р»

Россия, 634034, г. Томск,
ул. Нахимова д.8
тел. 8 (3822)90-22-53
tomsksoft@tomsksoft.ru
www.tomsksoft.ru

Справочное руководство по использованию библиотеки программных компонентов видеоэффектов (Effects SDK)

Содержание

Функциональные характеристики

Библиотека программных компонентов видео эффектов (Video Effects SDK) - это набор средств для разработки ПО, позволяющий встраивать видео эффекты в приложения для конечных пользователей.

Компоненты разработаны для пяти платформ: Windows, macOS, Android, iOS, WEB/WebRTC. На каждой из платформ есть платформозависимые реализации компонента, например, для запуска и работы нейронной сети обеспечивающей сегментацию фона используется OpenVino для Windows, CoreML для iOS и macOS, OnnxRuntime для Web и TensolFlow для Android, а для постобработки кадра на GPU используется DirectX для Windows, Metal для macOS, iOS, OpenGL ES для Android и WebGL для Web.

На всех платформах видео эффекты работают в режиме реального времени. Эффекты направлены на улучшение качества видео, повышения комфорта и эффективности проведения видеозвонков для различных сценариев их использования.

Таким образом продукт предназначен для компаний разработчиков и владельцев сервисов видео-чатов, видео-конференций, и других прикладных решений где присутствует функционал обмена он-лайн видео информацией с присутствием человека в кадре.

Инструкция по установке и эксплуатации

Windows SDK:

1. Требования

- Windows 10/11 (x64 platform)
- Windows x32 application supported
- Intel Core 6th gen & higher
- Direct3D 11
- C++ based SDK

2. Скачиваем и разархивируем файл (состоит из демо приложения x86/x64, демо файлов SDK, документации): https://mediasdk.ru/bin/Windows.zip

3. Запускаем демо версию приложения:

- Заходим в ./demo:
- Выбираем необходимую разрядность приложения (x86/x64) и запускаем файл: Effects SDK Demo.exe
- Выбираем камеру и включаем доступные эффекты.

4. Делаем тестовую интеграцию

- Копируем файлы из ./sdk в наш тестовый проект (в директорию где будет лежать модуль SDK, SDK подключается как динамическая библиотека)
- Инициализируем SDK как описано в документации ./docs
- Подаем кадры с камеры в SDK и получаем обработанные кадры в колбеке (детали есть в документации)
- Настраиваем какие эффекты должны применяться (детали есть в документации)

macOS SDK:

1. Требования

- MacOS 10.15 (11 for Arm)
- Mac ObjC wrapper: MacOS 10.15 (11 for Arm); Xcode 12+

2. Скачиваем и разархивируем файл (состоит из демо приложения (x64/arm), демо файлов SDK, документации): https://mediasdk.ru/bin/macOS.zip

3. Запускаем демо версию приложения:
- Заходим в ./demo
- Открываем gzip файл, после чего появится исполняемый файл, который запускаем.
- Выбираем камеру и включаем доступные эффекты.

4. Делаем тестовую интеграцию

- Копируем файлы из ./sdk в наш тестовый проект (в директорию где будет лежать модуль SDK, SDK подключается как динамическая библиотека)
- Инициализируем SDK как описано в документации ./docs
- Подаем кадры с камеры в SDK и получаем обработанные кадры в колбеке (детали есть в документации)
- Настраиваем какие эффекты должны применяться (детали есть в документации)

Android SDK:

1. Требования

- Java 1.8+
- Kotlin 1.4+
- Android Studio with a recommended version of at least 2021.1.1 (Bumblebee), or another compatible IDE

- Android 8.0 (SDK version 24) or higher for using Mediapipe features
- OpenGL ES 3.1 or higher for neural networks on GPU

2. Скачиваем и разархивируем файл (состоит из демо приложения (apk), демо файлов SDK, документации): https://mediasdk.ru/bin/Android.zip

3. Запускаем демо версию приложения:

- Заходим в ./demo:
- Открываем apk файл на Android устройстве.
- Выбираем камеру и включаем доступные эффекты.

4. Делаем тестовую интеграцию

- Копируем файлы из ./sdk в наш тестовый проект (в директорию где будет лежать модуль SDK, SDK подключается как динамическая библиотека)
- Инициализируем SDK как описано в документации ./docs
- Подаем кадры с камеры в SDK и получаем обработанные кадры в колбеке (детали есть в документации)
- Настраиваем какие эффекты должны применяться (детали есть в документации)

iOS SDK:

1. Требования

- iOS 13+
- iPhone 8+
- Xcode 12+

2. Скачиваем и разархивируем файл (состоит из демо приложения (исходные коды + SDK в виде xcframework), документации): https://mediasdk.ru/bin/iOS.zip

3. Запускаем демо версию приложения

- Копируем файлы из ./demo в отдельную папку (которая будет являться корнем нашего проекта)
- Устанавливаем Xcode версии 13.4.1 (необходимо сгенерировать сертификат для локальной разработки):
- Открываем нужный проект в Xcode.
- Подключаем iPhone/iPad по проводу к Mac устройству (мобильное устройство должно быть разблокировано).

- Собираем и запускаем проект на мобильном устройстве
- Разрешаем доступ к камере, по умолчанию запустится эффект замены фона.

4. Делаем тестовую интеграцию

- Копируем файл TSVB.xcframework из ./demo в наш тестовый проект (в директорию где будет лежать модуль SDK, SDK подключается как фреймворк)
- Собираем только через Xcode, добавляем TSVB.xcframework как зависимость
- Инициализируем SDK как описано в документации ./docs
- Подаем кадры с камеры в SDK и получаем обработанные кадры в колбеке (детали есть в документации)
- Настраиваем какие эффекты должны применяться (детали есть в документации)

Web SDK:

1. Требования

- Latest browsers Chrome, Safari, FireFox, Edge, Opera
- WebGL 1.0 and higher

2. Скачиваем и разархивируем файлы (состоит из демо приложения, файлов SDK, документации): https://mediasdk.ru/bin/Web.zip

3. Запускаем демо версию приложения:

- Запущенную демо версию можно посмотреть по ссылке: https://mediasdk.ru/sdk/demo

4. Делаем тестовую интеграцию

- Копируем файлы из ./sdk в отдельную директорию веб сервера (откуда будет загружаться библиотека)
    * Все файлы SDK должны загружаться по протоколу https
    * Все файлы SDK должны иметь заголовки: Access-Control-Allow-Origin:*
- Копируем файл index.html в корень веб сервера:
    * Сервер должен работать по https протоколу
- В файле index.html указываем путь к загружаемой библиотеке
- Открываем index.html через браузер
- В index.html настраиваем какие эффекты должны применяться (детали есть в документации)

Linux SDK:

1. Требования

- Ubuntu 18, 20, 22; Astra Linux Common Edition 2.12; RedOS
- На системе должен быть доступен XLib и OpenGL

2. Скачиваем и разархивируем файл (состоит из демо приложения, файлов SDK, документации): https://mediasdk.ru/bin/Linux.zip

3. Запускаем демо версию приложения:

- Заходим в ./demo:
- Запускаем Effects_SDK_Demo-x86_64.AppImage
- Выбираем камеру и включаем доступные эффекты.

4. Делаем тестовую интеграцию

- Копируем файлы из ./sdk в наш тестовый проект (в директорию где будет лежать модуль SDK, SDK подключается как динамическая библиотека)
- Инициализируем SDK как описано в документации ./docs
- Подаем кадры с камеры в SDK и получаем обработанные кадры в колбеке (детали есть в документации)
- Настраиваем какие эффекты должны применяться (детали есть в документации)

Руководство по использованию библиотеки для каждой из поддерживаемых платформ

Версия для Microsoft Windows

Технические детали

- SDK доступен для платформы Windows 10 x64.
- Также поддерживаются приложения x32, работающие на платформе x64.
- Предварительная/постобработка кадров может выполняться на процессоре или графическом процессоре.
- Вывод ML мог выполняться только на процессоре.

Функции

- Виртуальные фоны (поставить изображение в качестве фона)
- Размытие фона
- Улучшение внешности/сглаживание кожи
- Автоматическое кадрирование/Smart Zoom
- Автокоррекция цвета
- Цветокоррекция на основе примера

Сведения об использовании

Основным объектом SDK является экземпляр, реализующий ISDKFactory.
Используя экземпляр ISDKFactory, вы сможете подготовить кадры к обработке и настроить конвейер обработки (включить прозрачность, размытие, заменить фон и т. д.).

Как получить экземпляр tsvb::ISDKFactory

- Загрузите tsvb.dll с помощью функции **LoadLibrary()**.
- Получить адрес функции **createSDKFactory()** из dll. Приведите его к типу **::tsvb::pfnCreateSDKFactory**.
- Вызовите функцию **createSDKFactory()**, чтобы создать экземпляр объекта **::tsvb::ISDKFactory**.

```
::tsvb::ISDKFactory* createFactory()
{
   HMODULE hModule = ::LoadLibrary("tsvb");
   auto createSDKFactory = reinterpret_cast<::tsvb::pfnCreateSDKFactory>(
      GetProcAddress(hModule, "createSDKFactory")
   );
   return createSDKFactory();
}
```

Методы класса:
ISDKFactory::createFrameFactory() - создать экземпляр IFrameFactory.
ISDKFactory::createPipeline() - создать экземпляр IPipeline.

createSDKFactory() может возвращать NULL.

Управление памятью

Все классы, созданные с помощью SDK, реализуют **IRelease**. Интерфейс **IRelease** предоставляет метод **release()**, который освобождает память, выделенную для объекта. Метод **release()** следует вызывать явно, когда такой объект должен быть уничтожен.

```
::tsvb::ISDKFactory* sdkFactory = createFactory();
//some code
```

sdkFactory->release();

Не используйте оператор **delete** для объектов SDK.

Использование библиотеки

Подготовка:

- Создайте экземпляр **IFrameFactory**.
- Создайте экземпляр **IPipeline**.
- Включите размытие фона с помощью **IPipeline::enableBlurBackground()** или замену фона с помощью **IPipeline::enableReplaceBackground()**.
- Когда замена фона включена, вам нужно передать изображение, которое будет использоваться в качестве фона: **IReplacementController::setBackgroundImage()**

Обработка кадра:

- Поместите свой фрейм в **IFrame** с помощью **IFrameFactory::create()**.
- Обработать через **IPipeline::process()**.

Используйте отдельные экземпляры **IPipeline** для каждого видеопотока.

```
void initialize()
{
   ::tsvb::ISDKFactory* factory = createFactory();
   frameFactory = factory->createFrameFactory();
   pipeline = factory->createPipeline();
   pipeline->enableReplaceBackground(&replacementController);

   factory->release();
}

void release()
{
   frameFactory->release();
   frameFactory = nullptr;
   replacementController->release();
   replacementController = nullptr;
   pipeline->release();
   pipeline = nullptr;
}
```

Дополнительные сведения об использовании см. в: **Sample/BGReplacer.cpp**.

Описание классов

IFrameFactory

IFrameFactory можно создать, вызвав **ISDKFactory::createFrameFactory()**.

IFrameFactory::createBGRA() — создать **IFrame** из необработанных данных BGRA.
Параметры:

- **void* data** - указатель на данные BGRA.
- **unsigned int bytesPerLine** - количество байтов на строку фрейма.
- **unsigned int width** - количество пикселей по горизонтали.
- **unsigned int height** - количество пикселей по вертикали.
- **bool makeCopy** - если установлено значение true - **IFrame** будет копировать данные, иначе **IFrame** сохранит указатель на данные **(НЕ освобождайте данные во время их обработки)* *

IFrameFactory::createNV12() — создать **IFrame** из необработанных данных NV12.
Параметры:

- **void* yData** - указатель на данные компонента Y.
- **unsigned int yBytesPerLine** - количество байтов в одной строке матрицы компонентов Y.
- **void* uvData** - указатель на данные компонента UV.
- **unsigned int uvBytesPerLine** - количество байтов в одной строке матрицы компонента UV
- **unsigned int width** - количество пикселей по горизонтали.
- **unsigned int height** - количество пикселей по вертикали.
- **bool makeCopy** - то же поведение, что и для **IFrameFactory::createBGRA()**.

IFrameFactory::loadImage() - загрузить данные из изображения и создать **IFrame**. Верните NULL, если экземпляр не создан.
Параметры:

- **const char* utf8FilePath** - путь к файлу изображения. Путь должен быть в UTF-8.

IFrame

FrameFormat - формат представления данных.

- **bgra32** - формат с 8 битами на канал (32 бита на пиксель)
- **nv12** - формат NV12.

IFrame::frameFormat() - возвращаемый формат данных.

IFrame::width() - возвращает количество пикселей по горизонтали.
IFrame::height() - возвращает количество пикселей по вертикали.

IFrame::lock() - получить доступ к виртуальной памяти процесса. Возвращает интерфейс ILockedFrameData, который предоставляет возможность получать указатели на внутренние данные IFrame **(НЕ используйте IFrame до тех пор, пока ILockedFrameData не будет выпущен)**.

Параметры:

- **int frameLock** — может быть **FrameLock::read**, **FrameLock::write** или **FrameLock::readWrite**.

ILockedFrameData — сохранить доступ к данным внутри IFrame и вернуть указатели на эти данные.
Если он был получен с помощью **IFrame::lock()** с параметром **FrameLock::write** или **FrameLock::readWrite**, то изменения будут применены после освобождения **ILockedFrameData** .
Если оно было получено с помощью **IFrame::lock()** с параметром **FrameLock::read**, то не меняйте данные.

ILockedFrameData::dataPointer() - вернуть указатель на данные компонента.
Параметры:

- **int planarIndex** - зависит от формата фрейма. Для FrameFormat::bgra32 всегда следует использовать 0. Для FrameFormat::nv12 - 0 возвращает указатель на компонент Y, 1 возвращает указатель на компонент UV.

ILockedFrameData::bytesPerLine() - возвращает количество байтов в строке.
Параметры:

- **int planarIndex** - см. **ILockedFrameData::dataPointer()**.

IPipeline

PipelineErrorCode — коды ошибок для **IPipeline**.

- **ok** - успех
- **invalidArguemnt** - один или несколько аргументов неверны.
- **noFeaturesEnabled** - конвейер обработки не настроен.
- **engineInitializationError** - не удается инициализировать OpenVINO, аппаратное/программное обеспечение не поддерживается.
- **resourceAllocationError** - недостаточно памяти, места на диске и т.д.

IPipeline - конфигурация эффектов и процессора кадров. Используйте отдельные экземпляры для каждого видеопотока.

IPipeline::setConfiguration() — настроить конвейер, определить, что использовать для обработки изображений (см. **IPipelineConfiguration**), например, конвейер GPU или CPU. Этот

метод является необязательным.
Параметры:

- **const IPipelineConfiguration* config** — применяемая конфигурация.

IPipeline::copyConfiguration() — Возвращает копию текущей конфигурации. Вызывающий объект отвечает за освобождение возвращенного объекта.

IPipeline::copyDefaultConfiguration() — возвращает копию конфигурации по умолчанию. Вызывающий объект отвечает за освобождение возвращенного объекта.

IPipeline::enableBlurBackground() - включить размытие фона.
Параметры:

- **float blurPower** - мощность размытия от 0 до 1.

IPipeline::disableBackgroundBlur() - отключить размытие фона.

IPipeline::getBlurBackgroundState() — возвращает true, если включено размытие фона, иначе false.
Параметры:

- *float blurPower** - если не NULL, то сила размытия.

IPipeline::enableReplaceBackground() - включить замену фона, по умолчанию фон прозрачный. Пользовательское изображение для фона может быть установлено с помощью **IReplacementController::setBackgroundImage()**. Если включено размытие фона, пользовательское изображение также будет размыто.

Параметры:

- **IReplacementController** контроллер** - если не NULL, то будет новый экземпляр **IReplacementController**. Вызывающий объект отвечает за освобождение полученного экземпляра, когда он больше не нужен.

IPipeline::disableReplaceBackground() - отключить замену фона.

IPipeline::getReplaceBackgroundState() — возвращает true, если включена замена фона, иначе false.

IPipeline::enableBeautification() - включить украшение лица.
Примечание. Для macOS с украшением доступен только конвейер графического процессора.

IPipeline::disableBeautification() - отключить украшение лица.

IPipeline::setBeautitifcationLevel() - установить силу применения эффекта сглаживания лица
Параметры:

- **float level** - уровень может быть от 0 до 1. Большее число -> более заметный эффект сглаживания.

IPipeline::getBeautitifcationLevel() - получить текущую настройку силы применения эффекта сглаживания лица.

IPipeline::enableColorCorrection() - включить коррекцию цвета.
Примечание. Подготовка начинается асинхронно после обработки кадра, эффект может быть отложен.

IPipeline::disableColorCorrection() - отключить коррекцию цвета.

IPipeline::enableSmartZoom() - включить автоматический фрейминг (отслеживание лица и удержание заданного уровня приближения).

IPipeline::disableSmartZoom() - отключить автоматический фрейминг.

IPipeline::setSmartZoomLevel() - установить уровень приближения для автоматического фрейминга.

Параметры:

- **float level** - уровень может быть от 0 до 1. Определяет, какая площадь должна быть заполнена гранью. Большее число -> больше площади.

IPipeline::process() - Возвращает обработанный кадр того же формата, что и входной (со всеми примененными эффектами). В случае ошибки вернуть NULL.
Параметры:

- **const IFrame* input** - кадр для обработки.
- **PipelineError* error** - NULL или код ошибки.

IReplacementController::setBackgroundImage() - установить новое пользовательское фоновое изображение.
Параметры:

- **const IFrame* image** - произвольное изображение для фона, не освобождать при обработке.

IReplacementController::clearBackgroundImage() - очистить пользовательское фоновое изображение, фон будет прозрачным.

IPipelineConfiguration

Backend — серверная часть пайплайна.

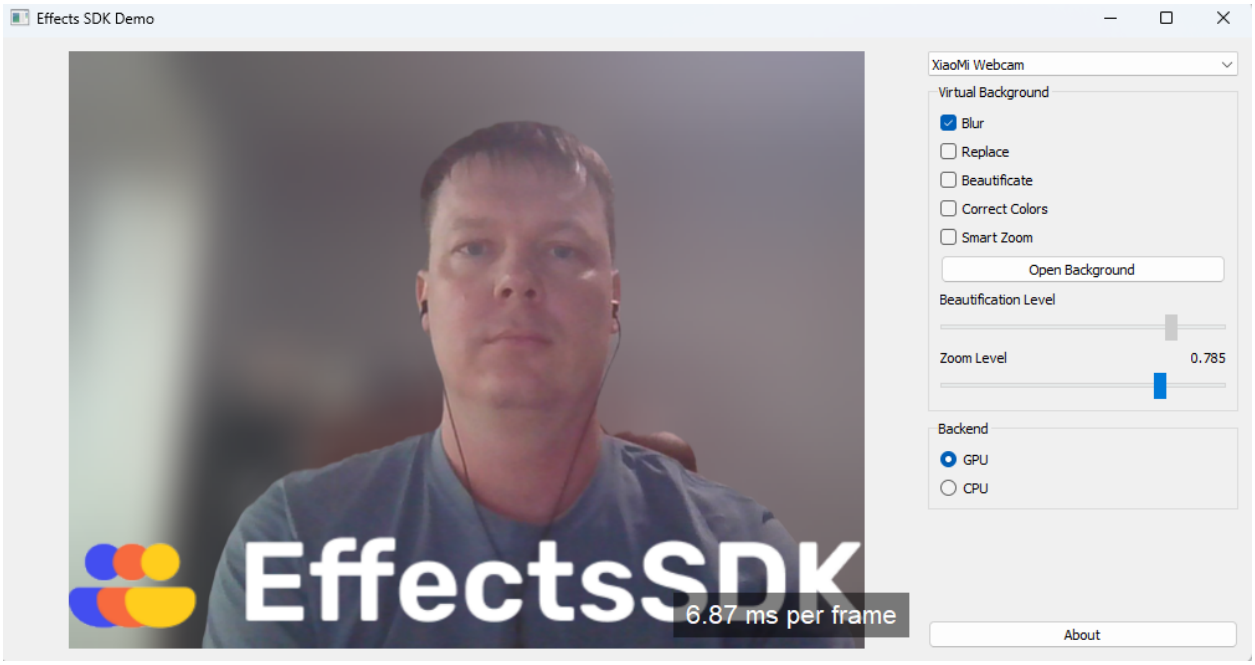
- **CPU** - конвейер на основе процессора.
- **GPU** - конвейер на базе видеокарты.

setBackend() — установить параметр бэкенда.
Параметры:

- **int backend** - должно быть одним из значений Backend.

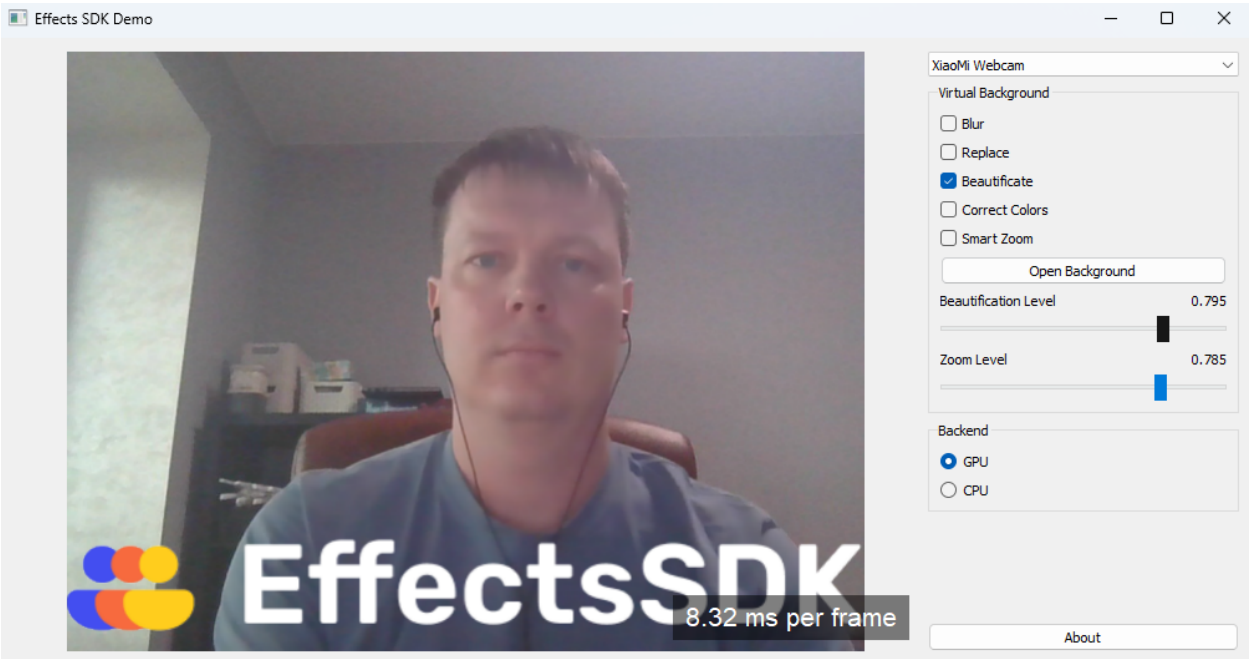
backend() - возвращает параметр бэкенда.

Демонстрационное приложение для Microsoft Windows

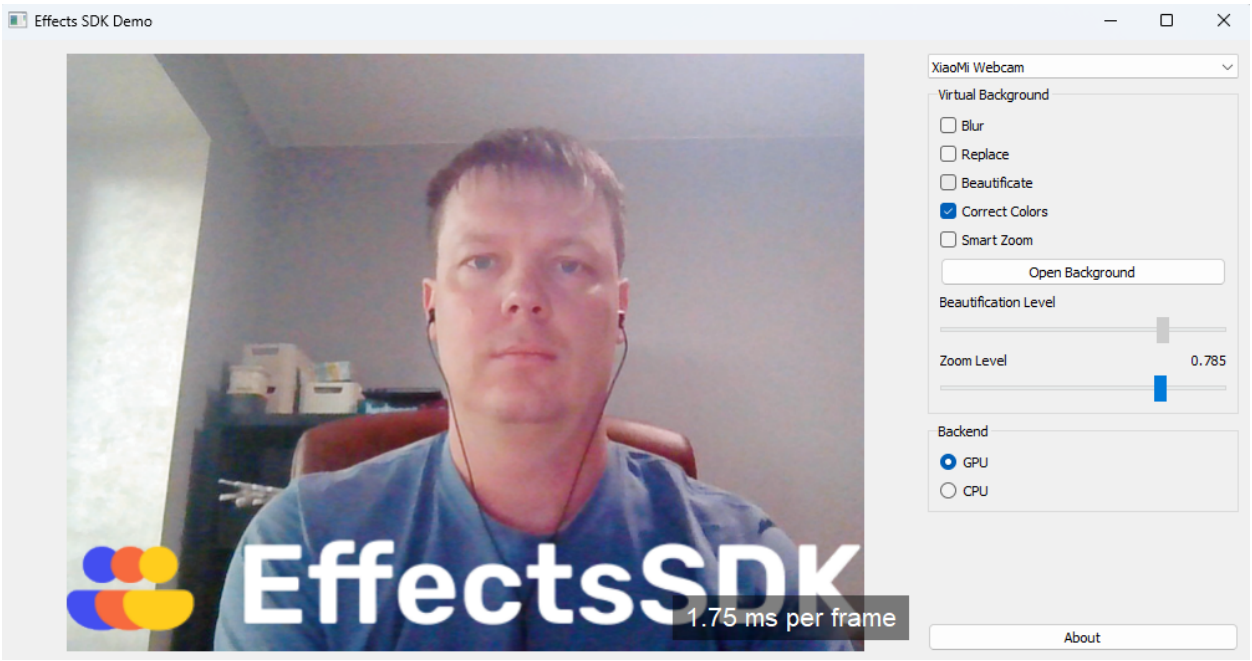


Blur - включить/отключить размытие фона за человеком.

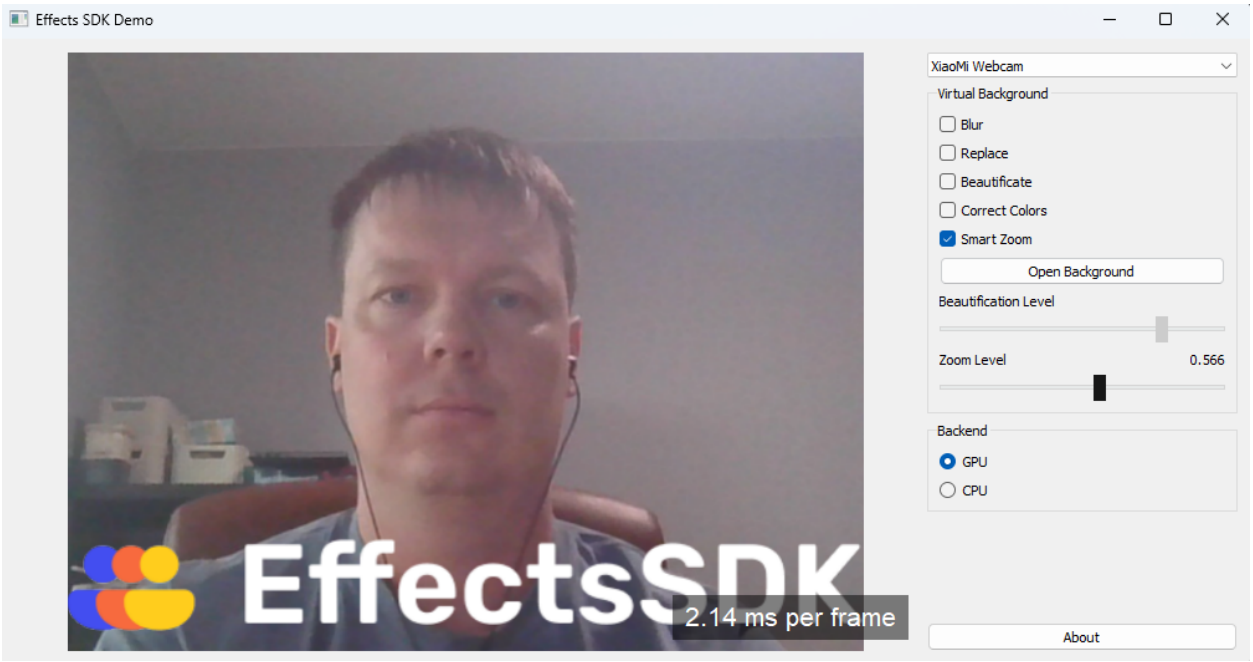
Replace - включить замену фона за человеком
Open Background - выбрать картинку на которую можно заменить фон, если не выбрана, то показываем картинку по умолчанию.



Beautification - включение/отключение сглаживания кожи лица
Beautification Level - настройка силы эффекта сглаживания лица



Correct Colors - включение/отключение автоматической коррекции цветов



Smart Zoom - включение/отключение автоматического фрейминга (автоматически следит за лицом человека в кадре и удерживает заданный уровень приближения)

Все эффекты могут быть использованы совместно/одновременно.

Также есть возможность переключения всех вычислений и отрисовки на CPU/GPU.

Версия для macOS

Технические детали

- SDK доступен для платформ macOS x64 и m1.
- Предварительная/постобработка кадров может выполняться на процессоре или графическом процессоре.
- Вывод ML мог выполняться только на процессоре.

Функции

- Виртуальные фоны (поставить изображение в качестве фона) - **реализовано**
- Размытие фона - **реализовано**
- Улучшение внешности/сглаживание кожи - **реализовано**
- Автоматическое кадрирование/Smart Zoom - **реализовано**
- Автокоррекция цвета - **реализовано**
- Цветокоррекция - **в процессе**

Сведения об использовании

Основным объектом SDK является экземпляр, реализующий ISDKFactory.
Используя экземпляр ISDKFactory, вы сможете подготовить кадры к обработке и настроить конвейер обработки (включить прозрачность, размытие, заменить фон и т. д.).

Как получить экземпляр tsvb::ISDKFactory

- Загрузите libtsvb.dylib с использованием функции **dlopen()**.
- Получить адрес функции **createSDKFactory()** из libtsvb.dylib. Приведите его к типу **::tsvb::pfnCreateSDKFactory**.
- Вызовите функцию **createSDKFactory()**, чтобы создать экземпляр объекта **::tsvb::ISDKFactory**.

```
::tsvb::ISDKFactory* createFactory()
{
	void* handle = dlopen("libtsvb.dylib", RTLD_LOCAL | RTLD_NOW);
	::vb_sdk::pfnCreateSDKFactory createFactory =
		reinterpret_cast<::vb_sdk::pfnCreateSDKFactory>(
			dlsym(_handle, "createSDKFactory")
		);

	return createSDKFactory();
}
```

Методы класса:
ISDKFactory::createFrameFactory() - создать экземпляр IFrameFactory.
ISDKFactory::createPipeline() - создать экземпляр IPipeline.

createSDKFactory() может возвращать NULL.

Управление памятью

Все классы, созданные с помощью SDK, реализуют **IRelease**. Интерфейс **IRelease** предоставляет метод **release()**, который освобождает память, выделенную для объекта. Метод **release()** следует вызывать явно, когда такой объект должен быть уничтожен.

```
::tsvb::ISDKFactory* sdkFactory = createFactory();
//some code
sdkFactory->release();
```

Не используйте оператор **delete** для объектов SDK.

Использование библиотеки

Подготовка:

- Создайте экземпляр **IFrameFactory**.
- Создайте экземпляр **IPipeline**.
- Включите размытие фона с помощью **IPipeline::enableBlurBackground()** или замену фона с помощью **IPipeline::enableReplaceBackground()**.
- Когда замена фона включена, вам нужно передать изображение, которое будет использоваться в качестве фона: **IReplacementController::setBackgroundImage()**

Обработка кадра:

- Поместите свой фрейм в **IFrame** с помощью **IFrameFactory::create()**.
- Обработать через **IPipeline::process()**.

Используйте отдельные экземпляры **IPipeline** для каждого видеопотока.

```
void initialize()
{
   ::tsvb::ISDKFactory* factory = createFactory();
   frameFactory = factory->createFrameFactory();
   pipeline = factory->createPipeline();
   pipeline->enableReplaceBackground(&replacementController);

   factory->release();
}

void release()
{
   frameFactory->release();
   frameFactory = nullptr;
   replacementController->release();
   replacementController = nullptr;
   pipeline->release();
   pipeline = nullptr;
}
```

Дополнительные сведения об использовании см. в: **Sample/BGReplacer.cpp**.

Описание классов

IFrameFactory

IFrameFactory можно создать, вызвав **ISDKFactory::createFrameFactory()**.

IFrameFactory::createBGRA() — создать **IFrame** из необработанных данных BGRA.
Параметры:

- **void* data** - указатель на данные BGRA.
- **unsigned int bytesPerLine** - количество байтов на строку фрейма.

- **unsigned int width** - количество пикселей по горизонтали.
- **unsigned int height** - количество пикселей по вертикали.
- **bool makeCopy** - если установлено значение true - **IFrame** будет копировать данные, иначе **IFrame** сохранит указатель на данные **(НЕ освобождайте данные во время их обработки)* *

IFrameFactory::createNV12() — создать **IFrame** из необработанных данных NV12.
Параметры:

- **void* yData** - указатель на данные компонента Y.
- **unsigned int yBytesPerLine** - количество байтов в одной строке матрицы компонентов Y.
- **void* uvData** - указатель на данные компонента UV.
- **unsigned int uvBytesPerLine** - количество байтов в одной строке матрицы компонента UV
- **unsigned int width** - количество пикселей по горизонтали.
- **unsigned int height** - количество пикселей по вертикали.
- **bool makeCopy** - то же поведение, что и для **IFrameFactory::createBGRA()**.

IFrameFactory::loadImage() - загружает данные из изображения и создает **IFrame**. Верните NULL, если экземпляр не создан.
Параметры:

- **const char* utf8FilePath** - путь к файлу изображения. Путь должен быть в UTF-8.

IFrame

FrameFormat - формат представления данных.

- **bgra32** - формат с 8 битами на канал (32 бита на пиксель)
- **nv12** - формат NV12.

IFrame::frameFormat() - возвращаемый формат данных.

IFrame::width() - возвращает количество пикселей по горизонтали.
IFrame::height() - возвращает количество пикселей по вертикали.

IFrame::lock() - получить доступ к виртуальной памяти процесса. Возвращает интерфейс ILockedFrameData, который предоставляет возможность получать указатели на внутренние данные IFrame **(НЕ используйте IFrame до тех пор, пока ILockedFrameData не будет выпущен)**.

Параметры:

- **int frameLock** — может быть **FrameLock::read**, **FrameLock::write** или **FrameLock::readWrite**.

ILockedFrameData — сохранить доступ к данным внутри IFrame и вернуть указатели на эти данные.
Если он был получен с помощью **IFrame::lock()** с параметром **FrameLock::write** или **FrameLock::readWrite**, то изменения будут применены после освобождения **ILockedFrameData** .
Если оно было получено с помощью **IFrame::lock()** с параметром **FrameLock::read**, то не меняйте данные.

ILockedFrameData::dataPointer() - вернуть указатель на данные компонента.
Параметры:

- **int planarIndex** - зависит от формата фрейма. Для FrameFormat::bgra32 всегда следует использовать 0. Для FrameFormat::nv12 - 0 возвращает указатель на компонент Y, 1 возвращает указатель на компонент UV.

ILockedFrameData::bytesPerLine() - возвращает количество байтов в строке.
Параметры:

- **int planarIndex** - см. **ILockedFrameData::dataPointer()**.

IPipeline

PipelineErrorCode — коды ошибок для **IPipeline**.

- **ok** - успех
- **invalidArguemnt** - один или несколько аргументов неверны.
- **noFeaturesEnabled** - конвейер обработки не настроен.
- **engineInitializationError** - не удается инициализировать OpenVINO, аппаратное/программное обеспечение не поддерживается.
- **resourceAllocationError** - недостаточно памяти, места на диске и т.д.

IPipeline - конфигурация эффектов и процессора кадров. Используйте отдельные экземпляры для каждого видеопотока.

IPipeline::setConfiguration() — настроить конвейер, определить, что использовать для обработки изображений (см. **IPipelineConfiguration**), например, конвейер GPU или CPU. Этот метод является необязательным.
Параметры:

- **const IPipelineConfiguration* config** — применяемая конфигурация.

IPipeline::copyConfiguration() — Возвращает копию текущей конфигурации. Вызывающий объект отвечает за освобождение возвращенного объекта.

IPipeline::copyDefaultConfiguration() — возвращает копию конфигурации по умолчанию. Вызывающий объект отвечает за освобождение возвращенного объекта.

IPipeline::enableBlurBackground() - включить размытие фона.
Параметры:

- **float blurPower** - мощность размытия от 0 до 1.

IPipeline::disableBackgroundBlur() - отключить размытие фона.

IPipeline::getBlurBackgroundState() — возвращает true, если включено размытие фона, иначе false.
Параметры:

- *float blurPower** - если не NULL, то сила размытия.

IPipeline::enableReplaceBackground() - включить замену фона, по умолчанию фон прозрачный. Пользовательское изображение для фона может быть установлено с помощью **IReplacementController::setBackgroundImage()**. Если включено размытие фона, пользовательское изображение также будет размыто.

Параметры:

- **IReplacementController** контроллер** - если не NULL, то будет новый экземпляр **IReplacementController**. Вызывающий объект отвечает за освобождение полученного экземпляра, когда он больше не нужен.

IPipeline::disableReplaceBackground() - отключить замену фона.

IPipeline::getReplaceBackgroundState() — возвращает true, если включена замена фона, иначе false.

IPipeline::enableBeautification() - включить эффект сглаживания лица.
Примечание. Эффект доступен только для конвейера графического процессора.

IPipeline::disableBeautification() - отключить эффект сглаживания лица.

IPipeline::setBeautitifcationLevel() - установить силу эффекта сглаживания лица.
Параметры:

- **float level** - уровень может быть от 0 до 1. Большее число -> более заметный эффект сглаживания лица.

IPipeline::getBeautitifcationLevel() - вернуть текущую силу эффекта сглаживания лица.

IPipeline::enableColorCorrection() - включить коррекцию цвета.
Примечание. Подготовка начинается асинхронно после обработки кадра, эффект может быть отложен.

IPipeline::disableColorCorrection() - отключить цветокоррекцию.

IPipeline::enableSmartZoom() - включить автоматический фрейминг (отслеживание лица и удержание заданного уровня приближения).

IPipeline::disableSmartZoom() - отключить автоматический фрейминг.

IPipeline::setSmartZoomLevel() - установить уровень приближения для автоматического фрейминга.

Параметры:

- **float level** - уровень может быть от 0 до 1. Определяет, какая площадь должна быть заполнена гранью. Большее число -> больше площади.

IPipeline::process() - вернуть обработанный кадр в том же формате, что и ввод (со всеми примененными эффектами). В случае ошибки, вернуть NULL.
Параметры:

- **const IFrame* input** - кадр для обработки.
- **PipelineError* error** - NULL или код ошибки.

IReplacementController::setBackgroundImage() - установить новое пользовательское фоновое изображение.
Параметры:

- **const IFrame* image** - произвольное изображение для фона, не освобождать при обработке.

IReplacementController::clearBackgroundImage() - очистить пользовательское фоновое изображение, фон будет прозрачным.

IPipelineConfiguration

Backend — серверная часть пайплайна.

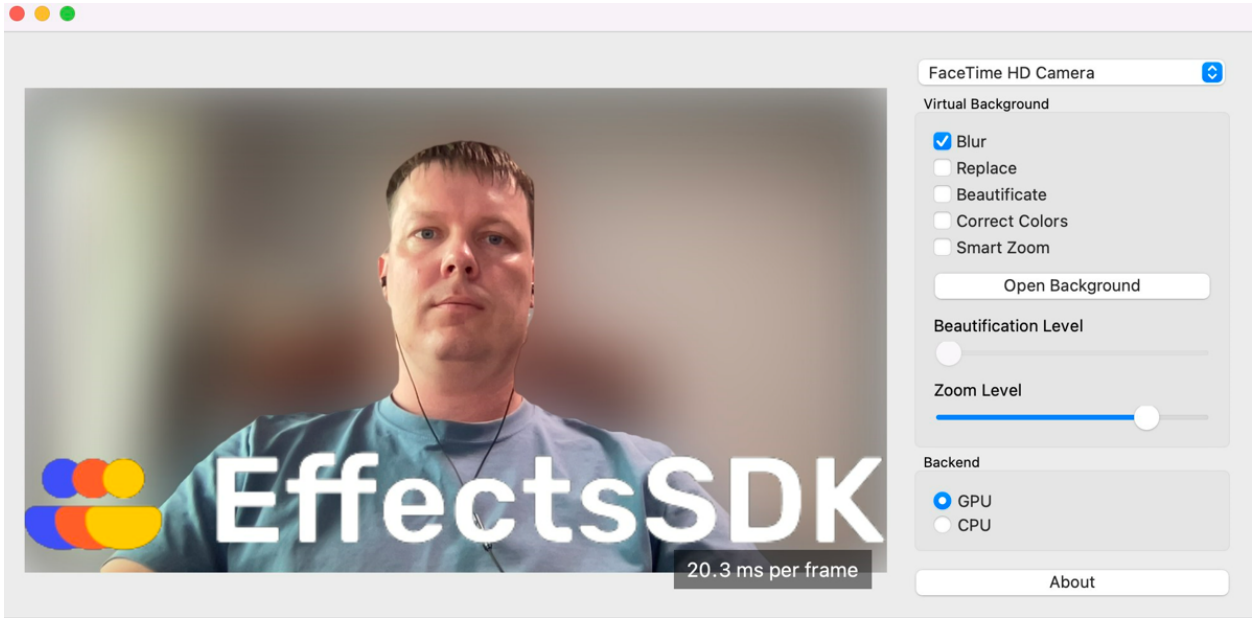
- **CPU** - конвейер на основе процессора.
- **GPU** - конвейер на базе GPU.

setBackend() — установить параметр бэкенда.
Параметры:

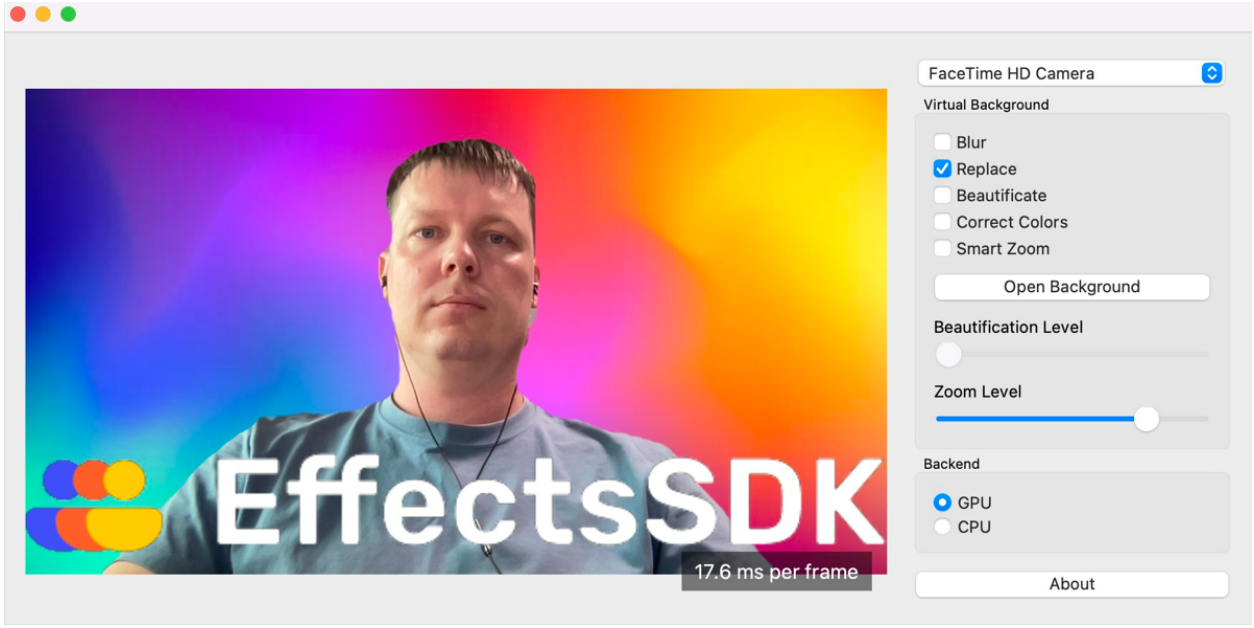
- **int backend** - должно быть одним из значений Backend.

backend() - возвращает параметр бэкенда.

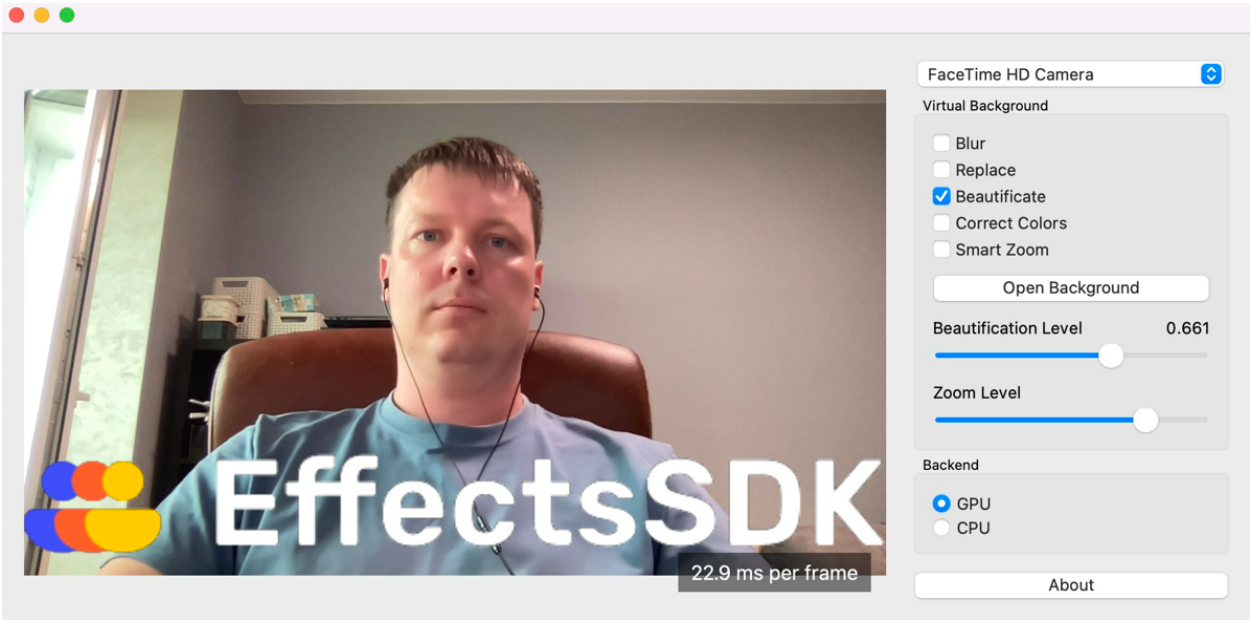
Демонстрационное приложение для macOS



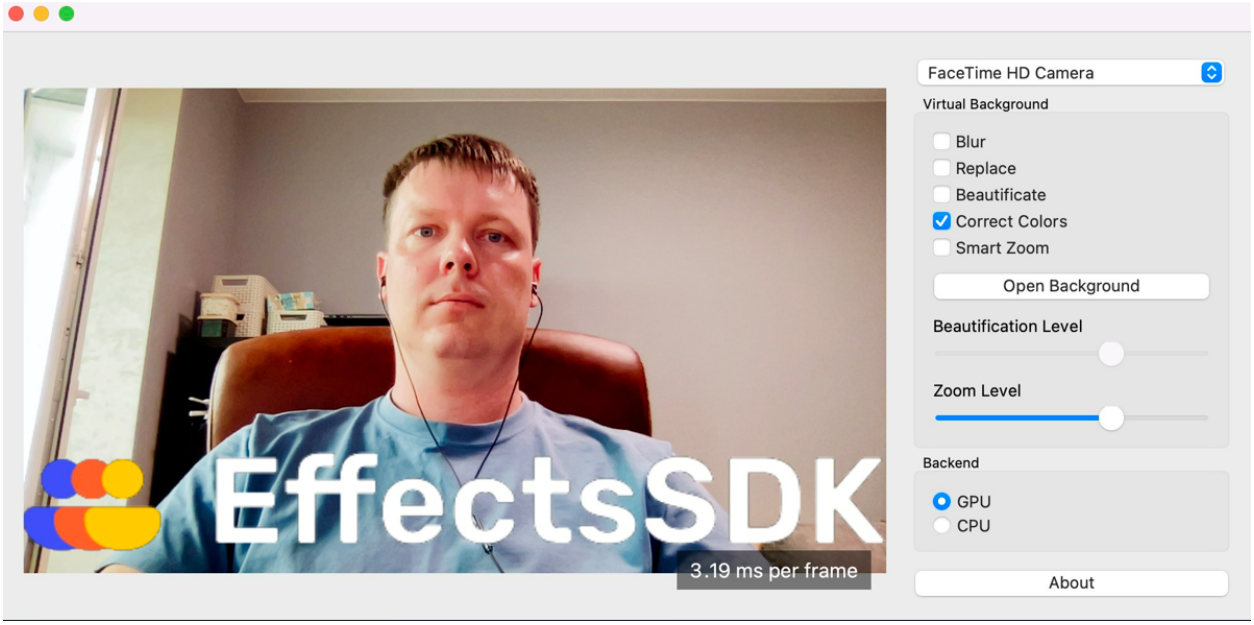
Blur - включить/отключить размытие фона за человеком.



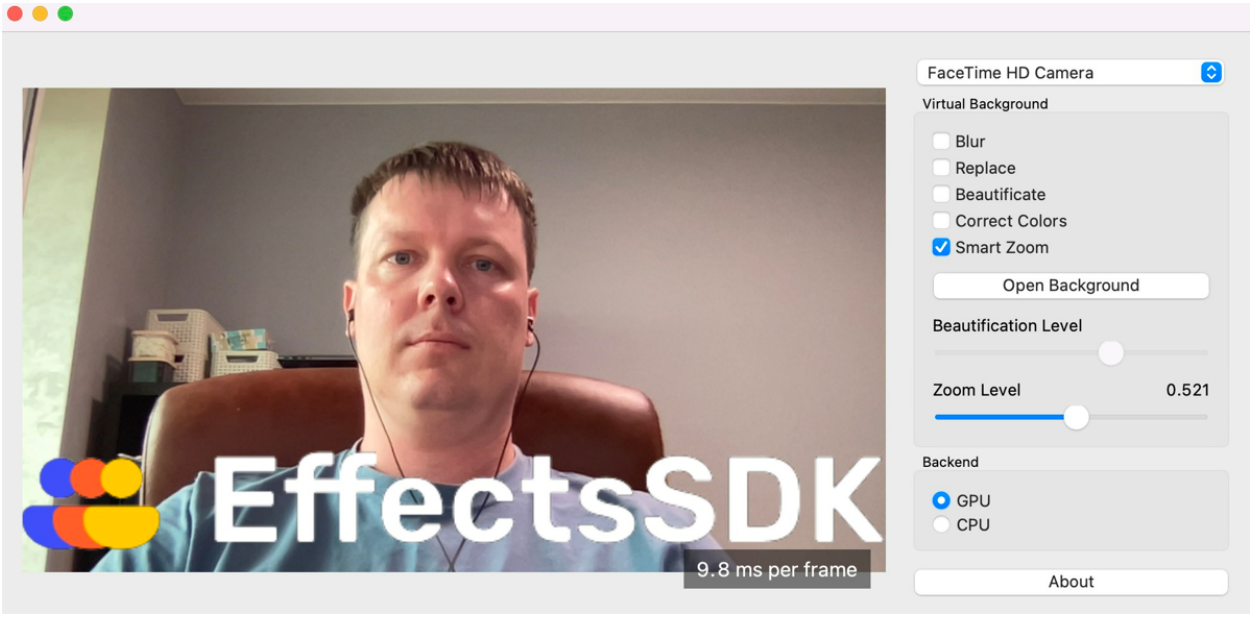
Replace - включить замену фона за человеком
Open Background - выбрать картинку на которую можно заменить фон, если не выбрана, то показываем картинку по умолчанию.



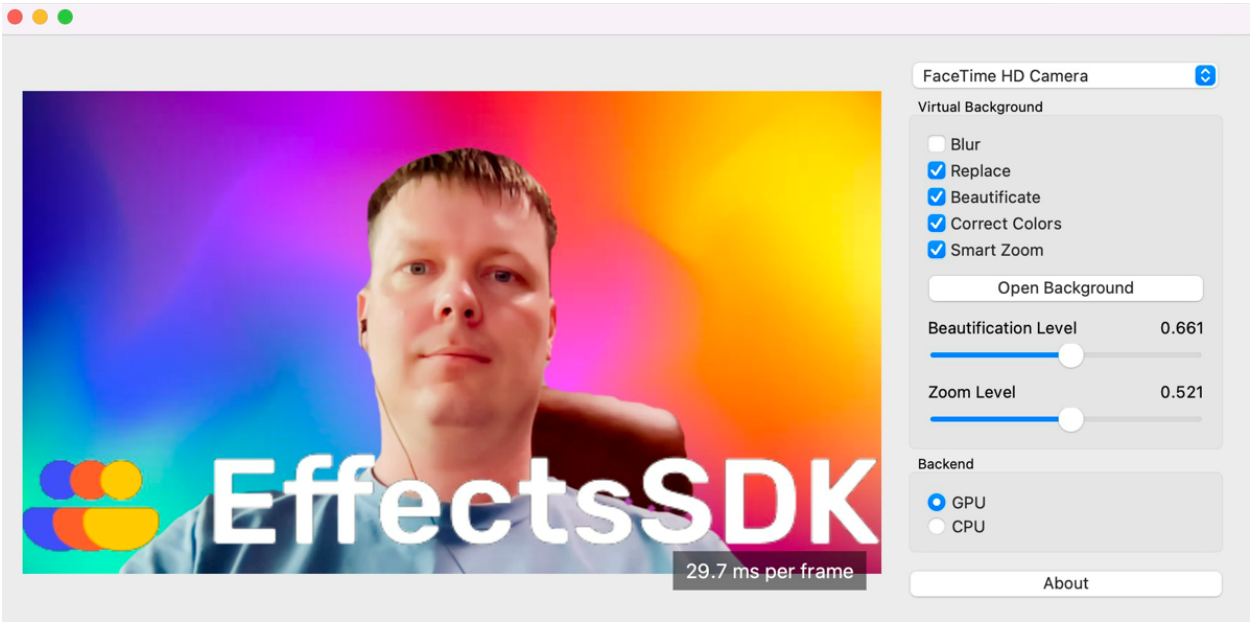
Beautification - включение/отключение сглаживания кожи лица
Beautification Level - настройка силы эффекта сглаживания лица



Correct Colors - включение/отключение автоматической коррекции цветов



Smart Zoom - включение/отключение автоматического фрейминга (автоматически следит за лицом человека в кадре и удерживает заданный уровень приближения)



Все эффекты могут быть использованы совместно/одновременно.

Также есть возможность переключения всех вычислений и отрисовки на CPU/GPU.

Версия для Linux

Технические детали

- SDK доступен для Linux (Ubuntu 18, 20, 22; Astra Linux Common Edition 2.12; RedOS)
- Предварительная/постобработка кадров может выполняться на процессоре или графическом процессоре.
- Вывод ML мог выполняться только на процессоре.

Функции

- Виртуальные фоны (поставить изображение в качестве фона) - **реализовано**
- Размытие фона - **реализовано**
- Улучшение/подкрашивание моей внешности - **реализовано**
- Автоматическое кадрирование/Smart Zoom - **реализовано**
- Автокоррекция цвета - **реализовано**
- Цветокоррекция - **в процессе**

Сведения об использовании

Основным объектом SDK является экземпляр, реализующий ISDKFactory.
Используя экземпляр ISDKFactory, вы сможете подготовить кадры к обработке и настроить конвейер обработки (включить прозрачность, размытие, заменить фон и т. д.).

Как получить экземпляр tsvb::ISDKFactory

- Загрузите libtsvb.so с использованием функции **dlopen**.
- Добавьте переменную env LD_LIBRARY_PATH, чтобы указать путь к библиотеке, например:
  экспорт LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/mnt/c/work/vbsdk-build-linux/lib
- Вызовите функцию **createSDKFactory()**, чтобы создать экземпляр объекта **::tsvb::ISDKFactory**.

```
::tsvb::ISDKFactory* createFactory()
{
	void* handle = dlopen("libtsvb.so", RTLD_NOW);
	::vb_sdk::pfnCreateSDKFactory createFactory =
		reinterpret_cast<::vb_sdk::pfnCreateSDKFactory>(
			dlsym(handle, "createSDKFactory")
		);

	return createSDKFactory();
}
```

Методы класса:
ISDKFactory::createFrameFactory() - создать экземпляр IFrameFactory.
ISDKFactory::createPipeline() - создать экземпляр IPipeline.

createSDKFactory() может возвращать NULL.

Управление памятью

Все классы, созданные с помощью SDK, реализуют **IRelease**. Интерфейс **IRelease** предоставляет метод **release()**, который освобождает память, выделенную для объекта. Метод **release()** следует вызывать явно, когда такой объект должен быть уничтожен.

```
::tsvb::ISDKFactory* sdkFactory = createFactory();
//some code
sdkFactory->release();
```

Не используйте оператор **delete** для объектов SDK.

Использование библиотеки

Подготовка:

- Создайте экземпляр **IFrameFactory**.
- Создайте экземпляр **IPipeline**.
- Включите размытие фона с помощью **IPipeline::enableBlurBackground()** или замену фона с помощью **IPipeline::enableReplaceBackground()**.
- Когда замена фона включена, вам нужно передать изображение, которое будет использоваться в качестве фона: **IReplacementController::setBackgroundImage()**

Обработка кадра:

- Поместите свой фрейм в **IFrame** с помощью **IFrameFactory::create()**.
- Обработать через **IPipeline::process()**.

Используйте отдельные экземпляры **IPipeline** для каждого видеопотока.

```
void initialize()
{
   ::tsvb::ISDKFactory* factory = createFactory();
   frameFactory = factory->createFrameFactory();
   pipeline = factory->createPipeline();
   pipeline->enableReplaceBackground(&replacementController);

   factory->release();
}

void release()
```

```
{
    frameFactory->release();
    frameFactory = nullptr;
    replacementController->release();
    replacementController = nullptr;
    pipeline->release();
    pipeline = nullptr;
}
```

Дополнительные сведения об использовании см. в: **Sample/BGReplacer.cpp**.

Описание классов

IFrameFactory

IFrameFactory можно создать, вызвав **ISDKFactory::createFrameFactory()**.

IFrameFactory::createBGRA() — создать **IFrame** из необработанных данных BGRA.
Параметры:

- **void* data** - указатель на данные BGRA.
- **unsigned int bytesPerLine** - количество байтов на строку фрейма.
- **unsigned int width** - количество пикселей по горизонтали.
- **unsigned int height** - количество пикселей по вертикали.
- **bool makeCopy** - если установлено значение true - **IFrame** будет копировать данные, иначе **IFrame** сохранит указатель на данные **(НЕ освобождайте данные во время их обработки)* *

IFrameFactory::createNV12() — создать **IFrame** из необработанных данных NV12.
Параметры:

- **void* yData** - указатель на данные компонента Y.
- **unsigned int yBytesPerLine** - количество байтов в одной строке матрицы компонентов Y.
- **void* uvData** - указатель на данные компонента UV.
- **unsigned int uvBytesPerLine** - количество байтов в одной строке матрицы компонента UV
- **unsigned int width** - количество пикселей по горизонтали.
- **unsigned int height** - количество пикселей по вертикали.
- **bool makeCopy** - то же поведение, что и для **IFrameFactory::createBGRA()**.

IFrameFactory::loadImage() - загрузить данные из изображения и создать **IFrame**. Верните NULL, если экземпляр не создан.
Параметры:

- **const char* utf8FilePath** - путь к файлу изображения. Путь должен быть в UTF-8.

IFrame

FrameFormat - формат представления данных.

- **bgra32** - формат с 8 битами на канал (32 бита на пиксель)
- **nv12** - формат NV12.

IFrame::frameFormat() - возвращаемый формат данных.

IFrame::width() - возвращает количество пикселей по горизонтали.
IFrame::height() - возвращает количество пикселей по вертикали.

IFrame::lock() - получить доступ к виртуальной памяти процесса. Возвращает интерфейс ILockedFrameData, который предоставляет возможность получать указатели на внутренние данные IFrame **(НЕ используйте IFrame до тех пор, пока ILockedFrameData не будет выпущен)**.

Параметры:

- **int frameLock** — может быть **FrameLock::read**, **FrameLock::write** или **FrameLock::readWrite**.

ILockedFrameData — сохранить доступ к данным внутри IFrame и вернуть указатели на эти данные.
Если он был получен с помощью **IFrame::lock()** с параметром **FrameLock::write** или **FrameLock::readWrite**, то изменения будут применены после освобождения **ILockedFrameData** .
Если оно было получено с помощью **IFrame::lock()** с параметром **FrameLock::read**, то не меняйте данные.

ILockedFrameData::dataPointer() - вернуть указатель на данные компонента.
Параметры:

- **int planarIndex** - зависит от формата фрейма. Для FrameFormat::bgra32 всегда следует использовать 0. Для FrameFormat::nv12 - 0 возвращает указатель на компонент Y, 1 возвращает указатель на компонент UV.

ILockedFrameData::bytesPerLine() - возвращает количество байтов в строке.
Параметры:

- **int planarIndex** - см. **ILockedFrameData::dataPointer()**.

IPipeline

PipelineErrorCode — коды ошибок для **IPipeline**.

- **ok** - успех

- **invalidArguemnt** - один или несколько аргументов неверны.
- **noFeaturesEnabled** - конвейер обработки не настроен.
- **engineInitializationError** - не удается инициализировать OpenVINO, аппаратное/программное обеспечение не поддерживается.
- **resourceAllocationError** - недостаточно памяти, места на диске и т.д.

IPipeline - конфигурация эффектов и процессора кадров. Используйте отдельные экземпляры для каждого видеопотока.

IPipeline::setConfiguration() — настроить конвейер, определить, что использовать для обработки изображений (см. **IPipelineConfiguration**), например, конвейер GPU или CPU. Этот метод является необязательным.
Параметры:

- **const IPipelineConfiguration* config** — применяемая конфигурация.

IPipeline::copyConfiguration() — Возвращает копию текущей конфигурации. Вызывающий объект отвечает за освобождение возвращенного объекта.

IPipeline::copyDefaultConfiguration() — возвращает копию конфигурации по умолчанию. Вызывающий объект отвечает за освобождение возвращенного объекта.

IPipeline::enableBlurBackground() - включить размытие фона.
Параметры:

- **float blurPower** - мощность размытия от 0 до 1.

IPipeline::disableBackgroundBlur() - отключить размытие фона.

IPipeline::getBlurBackgroundState() — возвращает true, если включено размытие фона, иначе false.
Параметры:

- *float blurPower** - если не NULL, то сила размытия.

IPipeline::enableReplaceBackground() - включить замену фона, по умолчанию фон прозрачный. Пользовательское изображение для фона может быть установлено с помощью **IReplacementController::setBackgroundImage()**. Если включено размытие фона, пользовательское изображение также будет размыто.

Параметры:

- **IReplacementController** контроллер** - если не NULL, то будет новый экземпляр **IReplacementController**. Вызывающий объект отвечает за освобождение полученного экземпляра, когда он больше не нужен.

IPipeline::disableReplaceBackground() - отключить замену фона.

IPipeline::getReplaceBackgroundState() — возвращает true, если включена замена фона, иначе false.

IPipeline::enableBeautification() - включить эффект сглаживания лица.

IPipeline::disableBeautification() - отключить эффект сглаживания лица.

IPipeline::setBeautitifcationLevel() - установить силу эффекта сглаживания лица.
Параметры:

- **Float level** - уровень может быть от 0 до 1. Большее число -> более заметный эффект сглаживания лица.

IPipeline::getBeautitifcationLevel() - вернуть текущую силу эффекта сглаживания лица.

IPipeline::enableColorCorrection() - включить коррекцию цвета.
Примечание. Подготовка начинается асинхронно после обработки кадра, эффект может быть отложен.

IPipeline::disableColorCorrection() - отключить цветокоррекцию.

IPipeline::enableSmartZoom() - включить автоматический фрейминг (отслеживание лица и удержание заданного уровня приближения).

IPipeline::disableSmartZoom() - отключить автоматический фрейминг.

IPipeline::setSmartZoomLevel() - установить уровень приближения для автоматического фрейминга.
Параметры:

- **float level** - уровень может быть от 0 до 1. Определяет, какая площадь должна быть заполнена гранью. Большее число -> больше площади.

IPipeline::process() - вернуть обработанный кадр в том же формате, что и входной (wco всеми примененными эффектами). В случае ошибки вернуть NULL.
Параметры:

- **const IFrame* input** - кадр для обработки.
- **PipelineError* error** - NULL или код ошибки.

IReplacementController::setBackgroundImage() - установить новое пользовательское фоновое изображение.
Параметры:

- **const IFrame* image** - произвольное изображение для фона, не освобождать при обработке.

IReplacementController::clearBackgroundImage() - очистить пользовательское фоновое изображение, фон будет прозрачным.

IPipelineConfiguration

Backend — серверная часть пайплайна.

- **CPU** - конвейер на основе процессора.
- **GPU** - конвейер на базе GPU.

setBackend() — установить параметр бэкенда.
Параметры:

- **int backend** - должно быть одним из значений Backend.

backend() - возвращает параметр бэкенда.

Демонстрационное приложение для Linux

Blur - включить/отключить размытие фона за человеком.

Replace - включить замену фона за человеком
Open Background - выбрать картинку на которую можно заменить фон, если не выбрана, то показываем картинку по умолчанию.

Beautification - включение/отключение сглаживания кожи лица
Beautification Level - настройка силы эффекта сглаживания лица

Smart Zoom - включение/отключение автоматического фрейминга (автоматически следит за лицом человека в кадре и удерживает заданный уровень приближения)

Correct Colors - включение/отключение автоматической коррекции цветов

Все эффекты могут быть использованы совместно/одновременно.

Также есть возможность переключения всех вычислений и отрисовки на CPU/GPU.

Версия для iOS

Технические детали

- SDK доступен для iOS 13 и новее.
- Предварительная/постобработка кадров может выполняться на процессоре или графическом процессоре.
- Вывод ML мог выполняться только на процессоре.

Функции

- Виртуальные фоны (поставить изображение в качестве фона) - **реализовано**
- Размытие фона - **реализовано**
- Улучшение/подкрашивание моей внешности - **реализовано**
- Автоматическое кадрирование/Smart Zoom - **реализовано**

- Автокоррекция цвета - **реализовано**
- Цветокоррекция - **в процессе**

Сведения об использовании

Точкой входа SDK является экземпляр TSVBSDKFactory.
Используя экземпляр TSVBSDKFactory, вы сможете подготовить кадры к обработке и настроить конвейер обработки (включить прозрачность, размытие, заменить фон и т. д.).

Использование

Подготовка:

- Создайте экземпляр **TSVBSDKFactory**.
- Создайте экземпляр **TSVBFrameFactory**, используя метод **newFrameFactory TSVBSDKFactory**.
- Создайте экземпляр **TSVBPipeline**, используя метод **newPipeline** из **TSVBSDKFactory**.
- Включите размытие фона с помощью метода **enableBlurBlurBlurBlurBackgroundWithPower:** или замену фона с помощью метода **enableReplaceBackground: TSVBPipeline**.
- Когда замена фона включена, вы можете передать изображение, которое будет использоваться в качестве фона: **TSVBReplacementController.background**

Обработка кадра:

- Поместите свой фрейм в **TSVBFrame**, используя **newFrameWithFormat:data:bytesPerLine:width:height:makeCopy:** метод **TSVBFrameFactory**.
- Обработайте его с помощью метода **process:error: TSVBPipeline**.

Используйте отдельные экземпляры **TSVBPipeline** для каждого видеопотока.

```
-(nullable id)init
{
    self = [super init];

    TSVBSDKFactory* sdkFactory = [TSVBSDKFactory new];
    _frameFactory = [sdkFactory newFrameFactory];
    _pipeline = [sdkFactory newPipeline];
    [_pipeline enableReplaceBackground:&_backgroundController];

    return self;
}
```

Подробнее об использовании см. в: **Sample/BackgroundReplacer.m**.

Описание классов

TSVBSDKFactory

-(nullable id<TSVBFrameFactory>)newFrameFactory;

Создает новый экземпляр **TSVBFrameFactory**.

-(nullable TSVBPipeline)newPipeline;

Создает новый экземпляр **TSVBPipeline**.

enum TSVBFrameFormat

- **TSVBFrameFormatRGBA** - формат RGBA с 8 битами на канал (32 бита на пиксель).
- **TSVBFrameFormatBGRA** - формат BGRA с 8 битами на канал (32 бита на пиксель).

enum TSVBFrameLock

- **TSVBFrameLockRead**
- **TSVBFrameLockWrite**
- **TSVBFrameLockReadWrite**

TSVBFrameFactory

```
-(id<TSVBFrame>)newFrameWithFormat:(TSVBFrameFormat)format
                              data:(void*)data
                     bytesPerLine:(unsigned int)bytesPerLine
                             width:(unsigned int)width
                            height:(unsigned int)height
                          makeCopy:(bool)makeCopy;
```

Создает **TSVBFrame** из необработанных данных RGBA или BGRA.

Параметры:

- **(TSVBFrameFormat)format** - формат сырых данных.
- **(void*)data** - указатель на необработанные данные.
- **(unsigned int)bytesPerLine** - количество байтов на строку фрейма.
- **(unsigned int)width** - количество пикселей по горизонтали.
- **(unsigned int)height** - количество пикселей по вертикали.
- **(bool)makeCopy** - если установлено в true - данные будут скопированы, иначе TSVBFrame сохранит указатель на данные (НЕ освобождайте данные во время их обработки).

-(id<TSVBGLFrame>)imageWithContentOfFile:(NSString*)filePath;

Загружает файл изображения и возвращает его как **TSVBFrame**. Если ARC отключен, используйте его в @autoreleasepool{ }

TSVBFrame

@property(nonatomic, readonly) unsigned int width;

Возвращает количество пикселей в горизонтальном направлении.

@property(nonatomic, только для чтения) unsigned int height;

Возвращает количество пикселей по вертикали.

@property(nonatomic, readonly) формат TSVBFrameFormat;

Возвращает формат кадра.

-(id<TSVBLockedFrameData>)lock:(TSVBFrameLock)lock;

Получает доступ к памяти кадра.
Возвращает протокол TSVBLockedFrameData, обеспечивающий возможность получения указателей на внутренние данные TSVBFrame (НЕ используйте TSVBFrame, пока TSVBLockedFrameData не будет освобожден).
Если ARC отключен, используйте его в @autoreleasepool{ }.

TSVBLockedFrameData

Сохраняет доступ к данным внутри TSVBFrame и возвращает указатели на эти данные.
Если он был получен с помощью **lock:TSVBFrameLockWrite** или **lock:TSVBFrameLockReadWrite** , то изменения будут применяться после того, как TSVBLockedFrameData будет выпущен.
Если он был получен с помощью **lock:TSVBFrameLockRead**, то данные не должны быть изменены.

-(void*)dataPointerOfPlanar:(int)index;

Возвращает указатель на планарные данные. Если **TSVBFrame**, созданный **newFrameWithFormat:data:bytesPerLine:width:height:makeCopy:**, где makeCopy был ложным, возвращает тот же указатель, который был передан
Параметры:

- **int index** - зависит от формата фрейма. Для TSVBFrameFormatRGBA или TSVBFrameFormatBGRA всегда следует использовать 0. Для TSVBFrameFormatNV12 - 0 возвращает указатель на компонент Y, 1 возвращает указатель на компонент UV.

-(unsigned int)bytesPerLineOfPlanar:(int)index;

Возвращает количество байтов в строке.
Параметры:

- **int index** - см. **dataPointerOfPlanar:**.

TSVBPipelineErrorCode - коды ошибок

- **TSVBPipelineErrorOk** - успех
- **TSVBPipelineErrorInvalidArgument** - один или несколько аргументов неверны.
- **TSVBPipelineErrorNoFeaturesEnabled** - конвейер обработки не настроен.
- **TSVBPipelineErrorEngineInitializationError** - не удается инициализировать OpenVINO, аппаратное/программное обеспечение не поддерживается.
- **TSVBPipelineErrorResourceAllocationError** - недостаточно памяти, места на диске и т. д.

TSVBPipeline

Настройка эффектов и кадрового процессора. Используйте отдельные экземпляры для каждого видеопотока.

-(TSVBPipelineError)setConfiguration:(id<TSVBPipelineConfiguration> _Nonnull)configuration;

Настраивает конвейер, определяет, что использовать для обработки изображений (см. **TSVBPipelineConfiguration**). Этот метод является необязательным.

-(id<TSVBPipelineConfiguration>)copyConfiguration;

Возвращает копию текущей конфигурации конвейера. Может использоваться для получения экземпляра TSVBPipelineConfiguration.

-(id<TSVBPipelineConfiguration>)copyDefaultConfiguration;

Возвращает копию конфигурации по умолчанию. Может использоваться для получения экземпляра TSVBPipelineConfiguration.

-(TSVBPipelineError)enabledBackgroundBlurWithPower:(float)power;

Включает размытие фона.
Параметры:

- **float power** - мощность размытия от 0 до 1.

-(void)disableBackgroundBlur

Отключает размытие фона.

-(TSVBPipelineError)enabledReplaceBackground:(id<ReplacementController>*)controller;

Включает замену фона, фон по умолчанию прозрачный. Пользовательское изображение для фона можно задать с помощью свойства background TSVBReplacementController.
Параметры:

- **(id<TSVBReplacementController>*)controller** — указатель на переменную для хранения экземпляра **TSVBReplacementController**. Может быть нулевым. Вызывающий объект отвечает за управление памятью для объектов вручную, если ARC отключен.

TSVBRReplacementController. Если включено размытие фона, пользовательское изображение также будет размыто.

-(void)disableReplaceBackground;

Отключает замену фона.

-(TSVBPipelineError)enabledBeautification;

Включает эффект сглаживания кожи лица.

-(void)disableBeautification;

Отключает эффект сглаживания кожи лица.

@property(nonatomic) float beautificationLevel;

Может быть от 0 до 1. Большее число -> более заметен эффект сглаживания кожи лица.

-(TSVBPipelineError)enabledColorCorrection;

Включает коррекцию цвета.
Примечание. Подготовка начинается асинхронно после обработки кадра, эффект может быть отложен.

-(void)disableColorCorrection;

Отключает коррекцию цвета.

-(TSVBPipelineError)enabledSmartZoom;

включить автоматический фрейминг (отслеживание лица и удержание заданного уровня приближения).

-(void)disableSmartZoom;

отключить автоматический фрейминг.

@property(nonatomic) float smartZoomLevel;

Параметры

- **float smartZoomLevel** - может быть от 0 до 1. Определяет, какая площадь должна быть заполнена лицом. Большее число -> больше площадь.

-(id<TSVBFrame>)process:(id<TSVBFrame>)frame error:(TSVBPipelineError*)error;

Возвращает обработанный кадр того же формата, что и входной (со всеми примененными эффектами). В случае ошибки возвращает NULL.
Параметры:

- **(id<TSVBFrame>)frame** - кадр для обработки.
- **(TSVBPipelineError*)error** - NULL или код ошибки.

TSVBRReplacementController

@property(nonatomic, readwrite, nullable)id<TSVBFrame> background

Содержит пользовательское изображение для замены фона. Если ноль, то обработка заменяет фон прозрачностью. Для сброса фона установите nil.

TSVBPipelineConfiguration

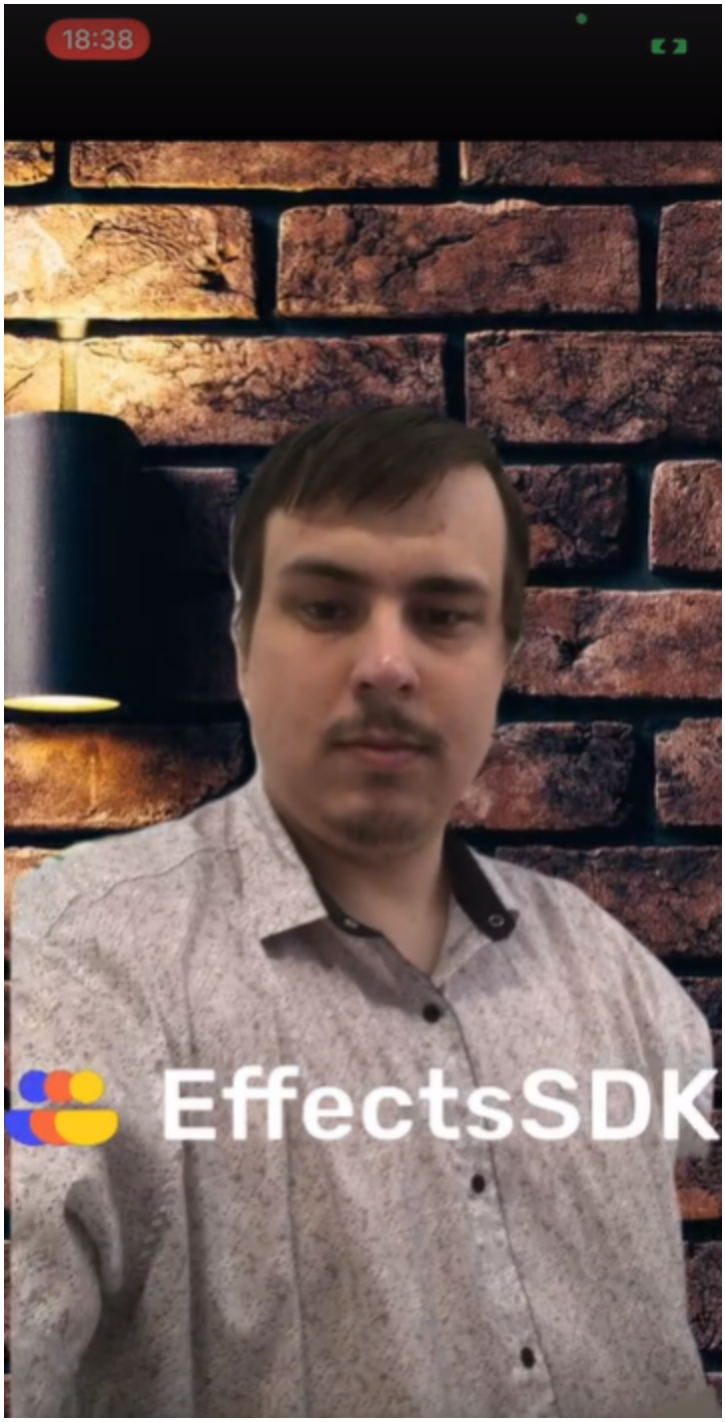
@property(nonatomic)enum TSVBBackend серверная часть

Определяет конвейер, выполняющий обработку изображений.

перечисление TSVBBackend

- **TSVBBackendCPU** - конвейер на основе процессора.
- **TSVBBackendGPU** - конвейер на основе графического процессора.

Демонстрационное приложение для iOS



В iOS демо версии нет никаких настроек, по умолчания включается режим замены фона на картинку.

Версия для Android

Подготовка

1. Добавьте зависимости в gradle файл
    1. implementation ‘com.google.flogger:flogger:0.6’
    2. implementation ‘com.google.flogger:flogger-system-backend:0.6’
    3. implementation ‘com.google.guava:guava:27.0.1-android’
2. Импортируйте AAR файл через Android Studio или добавьте gradle сценарий вручную
3. Вызовите EffectsSDK.initialize(context) метод в класее Application(или Activity) для загрузки библиотеки
4. Вызовите EffectsSDK.createSDKFactory(context) для получения SDKFactory экземпляра

Использование

1. Создайте экземпляр (Image/Camera)PipelineBuilder
2. Создайте экземпляр FrameFactory (в случае если Вы используете ImagePipeline)
3. Установите контекст для pipeline
4. Установите режим работы для pipeline (remove, replace, blur, no effects)
5. Установите дополнительные параметры (background image, etc)
6. Установите OnFrameAvailableListener для экземпляра Pipeline (если нужно получить Android Bitmap)
7. Установите surface для pipeline (если необходимо отрисовывать кадры сразу)

CameraPipeline обрабатывает вход камеры автоматически. Вызовите startPipeline() метод чтобы запустить этот процесс. Используйте setOutputSurface() метод чтобы связать Surface object с pipeline. Вы можете использовать onFrameAvailableListener чтобы получать Bitmap images из pipeline.

Пример кода

```
EffectsSDK.initialize(applicationContext)
private val sdkFactory = EffectsSDK.createSDKFactory()
private val pipeline = sdkFactory.createCameraPipelineBuilder()
        .setContext(activityLink)
        .setMode(ConfigMapper.map(sdkConfig.pipelineMode))
        .setBlurParams(sdkConfig.blurRadius, sdkConfig.blurQuality)
        .setBackground(sdkConfig.backgroundImage)
        .setGradingReference(sdkConfig.colorGradingReferenceImage)
        .setForegroundSize(sdkConfig.foregroundSize)
        .setSegmentationGap(sdkConfig.segmentationGap)
        .setFaceDetectionGap(sdkConfig.faceDetectionGap)
        .enableBeautification(sdkConfig.isBeautificationEnabled)
        .setColorCorrectionMode(ConfigMapper.map(sdkConfig.colorCorrectionMode))
        .setCamera(ConfigMapper.map(sdkConfig.camera))
        .setResolution(sdkConfig.resolution)
        .enableFPSCounter { view.onFPSCounterChanged(it) }
        .build()

pipeline.setOnFrameAvailableListener { bitmap ->
        //draw bitmap
```

```
}

pipeline.startPipeline()
```

Описание класса

EffectsSDK

initialize

fun initialize(context: Context)

Используйте метод для инициализации EffectsSDK. Контекст Application или Activity передается в качестве параметра.

Parameter name	Parameter type	Description
context	Context	Application context

createSDKFactory

fun createSDKFactory()

Создает экземпляр SDK factory.

getVersionName

fun getVersionName()

Возвращает версию SDK.

getCameraResolution

fun getCameraResolution(context: Context, camera: Camera)

Возвращает доступные разрешения входного видео с камеры (несортированный список).

Parameter name	Parameter type	Description
context	Context	Application context

camera	Camera	Front or back camera selection

FrameFactory

Объект может быть создан путем обращения к SDKFactory.createFrameFactory(). Этот клас используется для создания кадров.

createARGB

fun createARGB(bitmap: Bitmap)

Создает кадр из ARGB android bitmap.

Parameter name	Parameter type	Description
image	Bitmap	Android bitmap (ARGB_8888 format)

createYUV420

fun createYUV420(image: Image)

Создает кадр из YUVImage.

Parameter name	Parameter type	Description
image	Image	Instance of android.media.Image class (YUV_420_888 format)

Another formats will be added in next versions

Frame

frameFormat

fun frameFormat()

Возвращает информацию о формате.

width

fun width()

Возвращает количество пикселов в кадре по горизонтали.

height
fun height()

Возвращает количество пикселов в кадре по вертикали.

PipelineBuilder

setContext

fun setContext(act: Activity)

Привязывает android activity к pipeline.

Parameter name	Parameter type	Description
act	Activity	Android activity link.

setMode

fun setMode(mode: PipelineMode)

Устанавливает режимы pipeline (remove, replace, blur, no effects).

Parameter name	Parameter type	Description
mode	PipelineMode	Pipeline mode constant

setSegmentationMode

fun setSegmentationMode(mode: SegmentationMode)

Устанавливает одну из возможных вариантов моделей сегментации (portrait/landscape/auto).

Parameter name	Parameter type	Description

mode	SegmentationMode	Selected segmentation model

setBackground

fun setBackground(bitmap: Bitmap)

Устанавливает изображение для режима замены фона.

Parameter name	Parameter type	Description
image	Bitmap	Android bitmap (ARGB_8888 format)

setBlurParams

fun setBlurParams(radius: Float, quality: Float)

Включает функцию размытия фона. Если хотя бы один из параметров равен нулю, то функция не активируется.

Parameter name	Parameter type	Description
radius	Float	Values in [0…1]. More - slower, but stronger.
quality	Float	Values in [0…1]. More - slower, but stronger.

setForegroundSize

fun setForegroundSize(size: Int)

Устанавливает отступ от маски для сегментации.

Parameter name	Parameter type	Description

size	Int	Padding size. Values in [-4…4].

enableBeautification

fun enableBeautification(isEnabled: Boolean)

Включает/выключает функцию украшения лица.

Parameter name	Parameter type	Description
isEnabled	Boolean	Set true if you need to activate the beautification feature, else set false.

setColorCorrectionMode

fun setColorCorrectionMode(mode: ColorCorrectionMode)

Включает или отключает функцию цветокоррекции. Возможные значения параметра: NO_FILTER_MODE, COLOR_CORRECTION (with neural network), COLOR_GRADING(map color scheme between two images), PRESET_MODE (pre-defined filters).

Parameter name	Parameter type	Description
mode	ColorCorrectionMode	Type of color correction algorithm for pipeline

enableFpsCounter

fun enableFpsCounter(listener: FPSListener)

Устанавливает точку обратного вызова для fps.

Parameter name	Parameter type	Description

listener	FPSListener	The callback method has a fps value in parameters, you can handle it as you want.

setSegmentationGap

fun setSegmentationGap(gapSize: Int)

Устанавливает количество пропускаемых кадров при обработке pipeline. Например, если этот параметр равен 3, то сегментация будет происходить для каждого 4-го кадра.

Parameter name	Parameter type	Description
gapSize	Int	Number of frames to skip

setFaceDetectionGap

fun setFaceDetectionGap(gapSize: Int)

Устанавливает количество пропускаемых кадров для модели поиска лица. Например, если этот параметр равен 3, то модель будет обрабатывать каждый 4-й кадр.

Parameter name	Parameter type	Description
gapSize	Int	Number of frames to skip

setCamera

fun setCamera()

Устанавливает камеру для входа pipeline.

Parameter name	Parameter type	Description
cam	Camera	Select front or back camera for pipeline

setResolution
fun setResolution()

Устанавливает разрешение кадра для pipeline

Parameter name	Parameter type	Description
res	Size	Camera resolution

build
fun build()

Создает объект ImagePipeline.

BasePipeline
Этот интерфейс содержит общие методы для всех типов пайплайн.

getMode
fun getMode()

Возвращает текущий режим pipeline.

getBlurRadius
fun getBlurRadius()

Возвращает текущее значение радиуса для размытия фона.

getBlurQuality
fun getBlurQuality()

Возвращает текущее значение качества для размытия фона.

getForegroundSize
fun getForegroundSize()

Возвращает текущее значение отступа от маски для модели сегментации.

getSegmentationGap

fun getSegmentationGap()

Возвращает текущее значение пропуска кадров для модели сегментации.

getFaceDetectionGap
fun getFaceDetectionGap()

Возвращает текущее значение пропуска кадров для модели поиска лица.

isFlippedX
fun isFlippedX()

Возвращает ИСТИНА если кадр перевернут по оси X.

getZoomLevel
fun getZoomLevel()

Возвращает текущее значение уровня приближения для функции автоматического зума.

getSegmentationMode
fun getSegmentationMode()

Возвращает текущее значение режима модели сегментации (portrait/landscape/auto).

isBeautificationEnabled
fun isBeautificationEnabled()

Возвращает ИСТИНА если включен режим украшения лица.

setOutputSurface
fun setOutputSurface(surface: Surface)

Устанавливает поверхность для отрисовки кадров.

Parameter name	Parameter type	Description
surface	Surface	Surface instance

setFlipX
fun setFlipX(horizontalFlip: Bool)

Переворачивает кадр по оси X.

Parameter name	Parameter type	Description
horizontalFlip	Boolean	True, if you need flip by X axis and false if not

setOnFrameAvailableListener
fun setOnFrameAvailableListener(listener: OnFrameAvailableListener)

Устанавливает точку обратного вызова.

Parameter name	Parameter type	Description
listener	OnFrameAvailableListener	Callback interface instance.

setMode
fun setMode(mode: PipelineMode)

Устанавливает режим pipeline (remove, replace, blur, no effects).

Parameter name	Parameter type	Description
mode	PipelineMode	Pipeline mode constant

setZoomLevel
fun setZoomLevel(zoomLevel: Int)

Устанавливает процент приближения. Если передать значение меньше чем уже было установлено, то ничего не делает.

Parameter name	Parameter type	Description
zoomLevel	Int	Face size in percent

setBlurRadius
fun setBlurRadius(radius: Float)

Изменяет значение радиуса для фильтра размытия фона.

Parameter name	Parameter type	Description
radius	Float	Values in [0…1]. More - slower, but stronger.

setBlurQuality
fun setBlurQuality(quality: Float)

Устанавливает значение силы размытия для размытия фона.

Parameter name	Parameter type	Description
quality	Float	Values in [0…1]. More - slower, but stronger.

setForegroundSize
fun setForegroundSize(size: Int)

Устанавливает отступ для маски сегментации.

Parameter name	Parameter type	Description

size	Int	Padding size. Values in [-4…4].

setSegmentationGap

fun setSegmentationGap(gapSize: Int)

Устанавливает количество кадров для модели сегментации. Например, если параметр равен 3, то сегментация будет выполняться на каждом 4-м кадре.

Parameter name	Parameter type	Description
gapSize	Int	Number of frames to skip

setFaceDetectionGap

fun setFaceDetectionGap(gapSize: Int)

Устанавливает количество пропускаемых кадров для модели поиска лица. Например, если параметр равен 3, то поиск лица будет выполняться на каждом 4-м кадре.

Parameter name	Parameter type	Description
gapSize	Int	Number of frames to skip

enableBeautification

fun enableBeautification(isEnabled: Boolean)

Включает/выключает режим украшения лица.

Parameter name	Parameter type	Description

isEnabled	Boolean	Set true if you need to activate the beautification feature, else set false.

setColorCorrectionMode

fun setColorCorrectionMode(mode: ColorCorrectionMode)

Включает/выключает режим цветокорректции. Доступные режимы: NO_FILTER_MODE, COLOR_CORRECTION (with neural network), COLOR_GRADING(map color cheme between two images), PRESET_MODE (pre-defined filters).

Parameter name	Parameter type	Description
mode	ColorCorrectionMode	Type of color correction algorhitm for pipeline

setColorGradingReferenceImage

fun setColorGradingReferenceImage(bitmap: Bitmap)

Устанавливает эталонное изображение для режима коррекции цвета

Parameter name	Parameter type	Description
bitmap	Bitmap	Reference image bitmap

setImageFilter

fun setImageFilter(filter: ImageFilterPreset)

Устанавливает фильтр для функции цветокоррекции.

Parameter name	Parameter type	Description

filter	ImageFilterPreset	Lut for image

setBeautificationPower

fun setBeautificationPower(power: Int)

Устанавливает степень применения эффекта “Украшение лица”.

Parameter name	Parameter type	Description
power	Int	Value from [0…100]

setBackground

fun setBackground(bitmap: Bitmap)

Устанавливает изображения для режима замены фона.

Parameter name	Parameter type	Description
image	Bitmap	Android bitmap (ARGB_8888 format)

setSegmentationMode

fun setSegmentationMode(mode: SegmentationMode)

Устанавливает модель для сегментации (portrait/landscape/auto).

Parameter name	Parameter type	Description
mode	SegmentationMode	Selected segmentation model

release

fun release()

Закрывает pipeline и освобождает занимаемую память.

CameraPipeline
Этот pipeline используется для обработки входных данных с камеры.

startPipeline
fun startPipeline()

Запускает pipeline. Экземпляр Camera стартует после вызова этого метода.

setResolution
fun setResolution()

Устанавливает разрешение камеры для pipeline

Parameter name	Parameter type	Description
res	Size	Camera resolution

ImagePipeline
Этот pipeline используется для обработки произвольных данных из любого источника.

process
fun process(frame: Frame)

Запускает кадр на обработку в pipeline.

Parameter name	Parameter type	Description
frame	Frame	EffectsSDK frame.

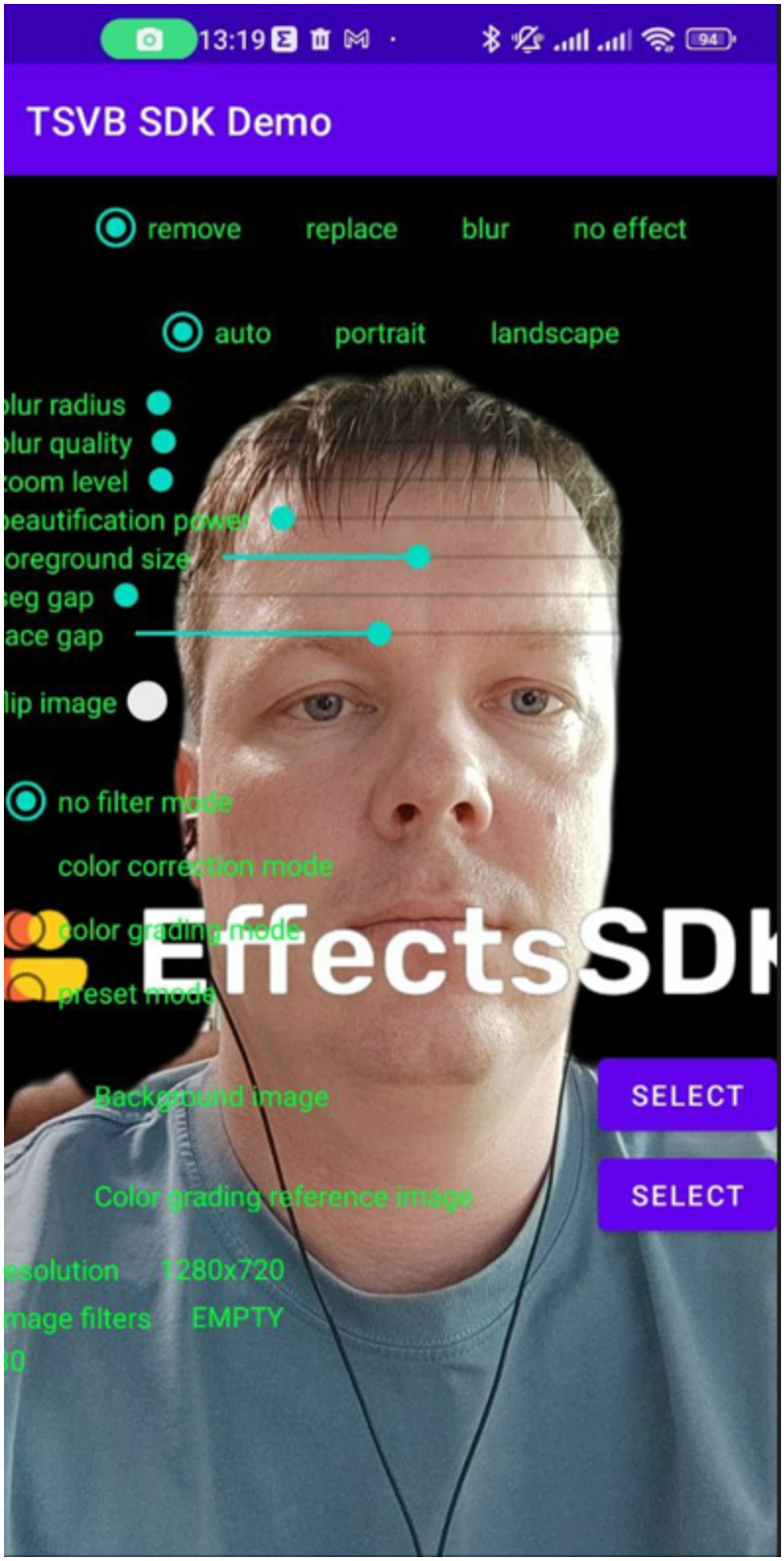
OnFrameAvailableListener
Используйте этот интерфейс для получения Bitmap objects из pipeline.

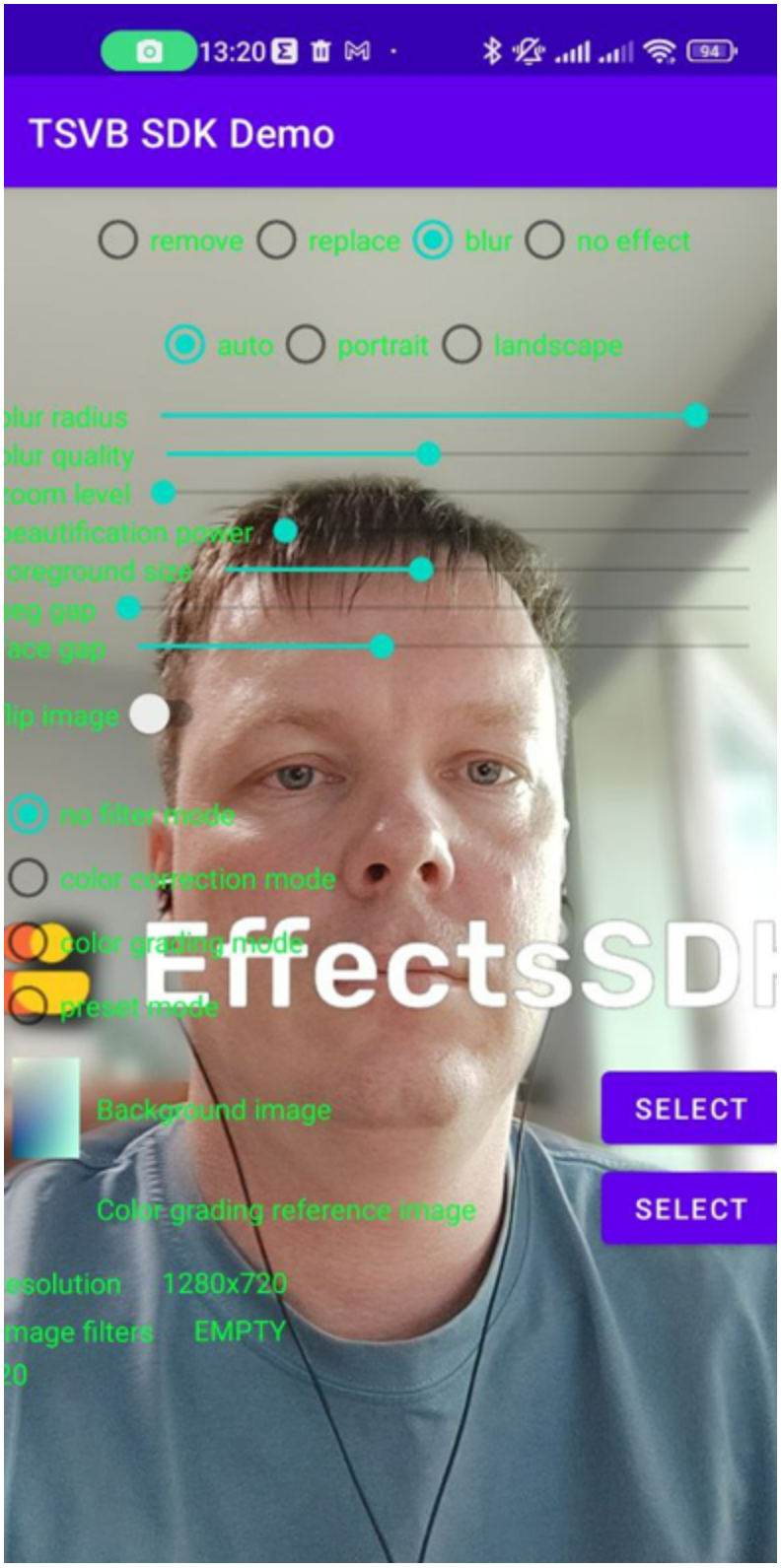
onNewFrame
fun onNewFrame(bitmap: Bitmap)

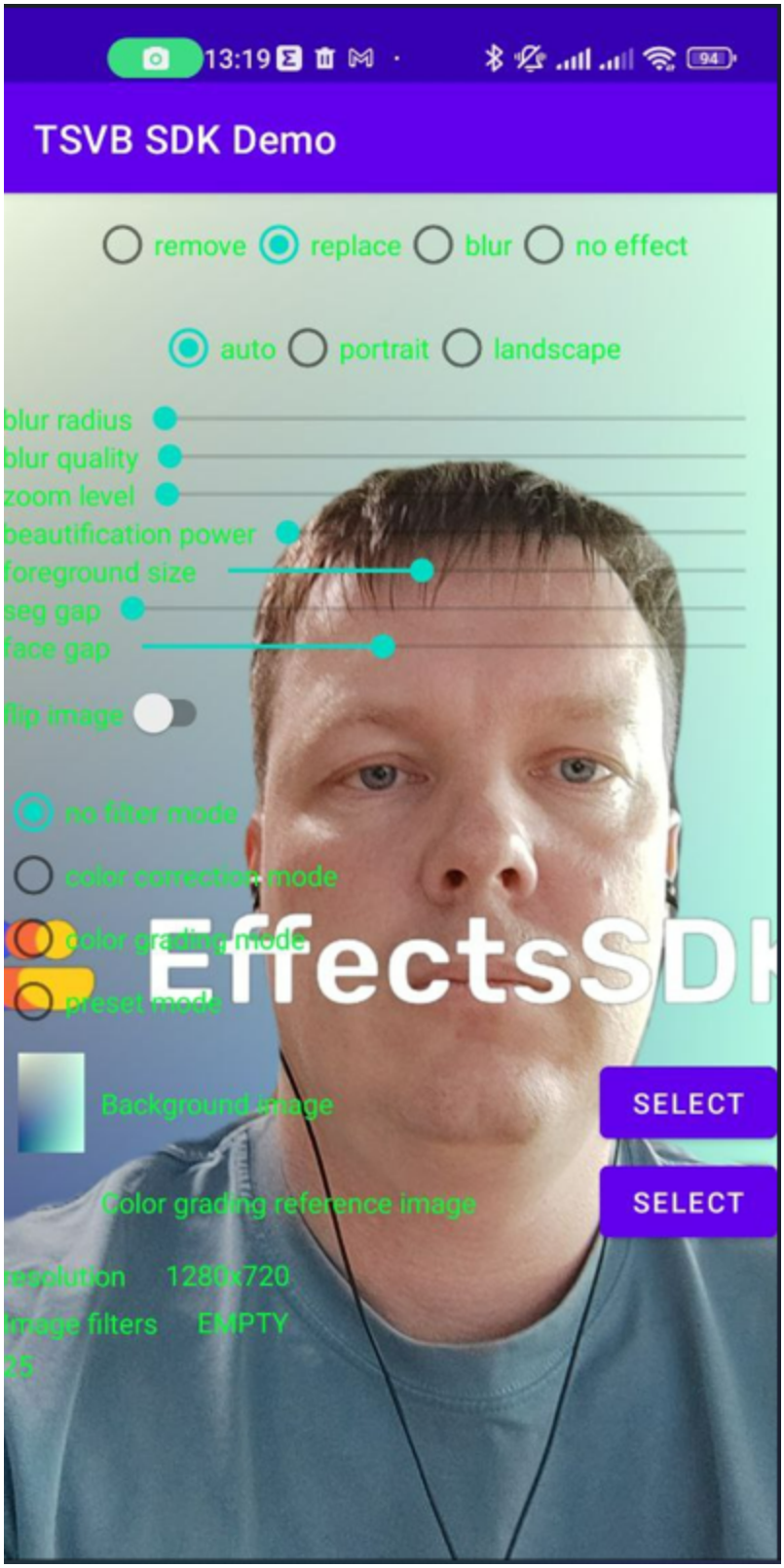
Результат работы pipeline будет записан сюда как android bitmap.

Parameter name	Parameter type	Description
bitmap	Bitmap	ARGB android bitmap with all effects.

Демонстрационное приложение для Android


13:19
TSVB SDK Demo
remove
replace
blur
no effect
auto
portrait
landscape
lur radius
lur quality
oom level
eautification power
oreground size
eg gap
ace gap
lip image
no filter mode
color correction mode
color grading mode
preset mode
EffectsSDK
Background image
SELECT
Color grading reference image
SELECT
esolution 1280x720
mage filters EMPTY
0


13:20
94
TSVB SDK Demo
remove
replace
blur
no effect
auto
portrait
landscape
lur radius
lur quality
oom level
eautification power
oreground size
eg gap
ce gap
lip image
no filter mode
color correction mode
color grading mode
preset mode
Background image
SELECT
Color grading reference image
SELECT
esolution 1280x720
mage filters EMPTY
20



Remove - удалить фон за человеком

Replace - заменить фон за человеком, для замены необходимо выбрать картинку для фона (Background Image -> Select)

Blur - включить размытие фона за человеком

No effect - отключить все эффекты

Auto/portrait/landscape - выбрать режим применения модели сегментации (для портретной и для альбомной картинки используется разные модели сегментации).

Blur radius - настройка радиуса для гауссово размытия фона

Blur quality - настройка силы для гауссово размытия фона

Zoom level - уровень приближения для автоматического фрейминга (удерживает лицо в кадра с предустановленным приближение)

Beautification power - сила применения эффекта сглаживания кожи лица

Foreground size - уровень отступа маски сегментации от человека (может появиться эффект ауры)

Seg gap - указание сколько кадров пропускать в режиме сегментации (режим реализован для слабых устройств)

Face gap - указание сколько кадров пропускать при поиске лица в кадре (больше пропусков - менее отзывчивый smart zoom)

Flip-image - переворот картинки по горизонтали

Color correction mode - включить автоматическую коррекцию цвета

Color grading mode - включить цветокоррекцию на основе шаблонного изображения (необходимо выбрать референсную картинку Color grading reference image -> Select)

Версия для Web

Технические подробности

- SDK имеет 3 предустановки скорости/качества (разные модели сегментации).
- Для улучшения вывода FPS SDK имеет возможность пропускать кадры.
- SSL сертификат для получения MediaStream через веб-браузер.

- Поддержка браузером WebGL 1.0

Функции

- Виртуальные фоны (поставить изображение в качестве фона)
- Размытие фона
- Улучшение/подкрашивание моей внешности
- Автоматическое кадрирование/Smart Zoom
- Автокоррекция цвета
- Цветокоррекция

Описание вызовов для JS

Constructors

Эту документацию можно получить в html виде по ссылке
https://mediasdk.ru/sdk/web/docs/classes/tsvb.html
constructor

- new tsvb(customer_id: string, inference?: any): tsvb
  - Defined in tsvb.ts:51

  Initiation of main SDK instance.

  Parameters
  - customer_id: string
    the unique customer identifier provided by SDK vendor.
  - inference: any = null

  Returns tsvb

Properties

components
components: {} = ...
Defined in tsvb.ts:29
Type declaration
- [key: string]: Component

customer_id
customer_id: string
Defined in tsvb.ts:36
Уникальный идентификатор пользователя, который выдается владельцем SDK, его можно получить бесплатно через сайт https://mediasdk.ru.
Optional onReady
onReady?: (() => void)
Defined in tsvb.ts:19

Type declaration
(): void
Returns void
Static HEIGHT
HEIGHT: number = 480
Defined in tsvb.ts:39
Static WIDTH
WIDTH: number = 640
Defined in tsvb.ts:38
Static Optional onAuthRequested
onAuthRequested?: ((url: string, payload: Object) => Promise<string>)
Defined in tsvb.ts:40
Type declaration
(url: string, payload: Object): Promise<string>
Parameters
- url: string
- payload: Object

Returns Promise<string>

Methods

addComponent

- addComponent<K>(c: ClassType<K>, id: string): void
  - Defined in tsvb.ts:782

  Type Parameters
  - K extends "overlay_screen" | "watermark" | "lower_third_1"

  Parameters
  - c: ClassType<K>
  - id: string

  Returns void

clear

- clear(): boolean
  Defined in tsvb.ts:701

  Clear output stream.

  Returns boolean

clearBackground

- clearBackground(): boolean
  Defined in tsvb.ts:438

  Disable background effect. As a background will be shown original video.

  Returns boolean

clearBlur

- clearBlur(): boolean
  Defined in tsvb.ts:413

  Disable blur of the background.

  Returns boolean

config

- config(config: any): void
  - Defined in tsvb.ts:94

  Ability to configure sdk execution environment

  remarks

proxy - configuration specify if segmentation should be working in separate worker thread (not in main UI thread), default value is true
wasmPaths - specify the paths for wasm binaries as a backend of ML execution for onnxruntime

  example

```
config = {
  api_url: 'url',
  model_url: 'url',
  proxy: true/false,
  stats: true/false,
  wasmPaths: {
    'ort-wasm.wasm': 'url',
    'ort-wasm-simd.wasm': 'url',
    'ort-wasm-threaded.wasm': 'url',
    'ort-wasm-simd-threaded.wasm': 'url'
  }
}
```

  Parameters
  - config: any

    configuration object

  Returns void

createComponent

- createComponent<K>(arg: K extends keyof OptionsMap ? ComponentArguments<K> : Omit<ComponentArguments<K>, "options">): ClassType<K>
  - Defined in tsvb.ts:777

  Type Parameters
  - K extends "overlay_screen" | "watermark" | "lower_third_1"

  Parameters
  - arg: K extends keyof OptionsMap ? ComponentArguments<K> : Omit<ComponentArguments<K>, "options">

  Returns ClassType<K>

disableBeautification

- disableBeautification(): boolean

    Defined in tsvb.ts:333

  Disable beautification effect.

  Returns boolean

disableColorCorrector

- disableColorCorrector(): boolean

    Defined in tsvb.ts:640

Disable color-corrector effect.
Returns boolean

disableFrameSkipping

- disableFrameSkipping(): boolean
  Defined in tsvb.ts:462
  Disable Frame Skipping - segmentation will be running on every video frame.
  FrameSkipping disabled by default.
  Returns boolean

disablePipelineSkipping

- disablePipelineSkipping(): void
  Defined in tsvb.ts:768
  Returns void

disableSmartZoom

- disableSmartZoom(): boolean
  Defined in tsvb.ts:616
  Disable smart-zoom effect.
  Returns boolean

enableBeautification

- enableBeautification(): boolean
  Defined in tsvb.ts:321
  Enable beautification effect.
  Returns boolean

enableColorCorrector

- enableColorCorrector(): boolean
  Defined in tsvb.ts:628
  Enable color-correction effect.
  Returns boolean

enableFrameSkipping

- enableFrameSkipping(): boolean
  Defined in tsvb.ts:450
  Enable Frame Skipping - segmentation will be running on every second frame, this will increase FPS but brings some motion trail
  Returns boolean

enablePipelineSkipping

- enablePipelineSkipping(): void
  Defined in tsvb.ts:764
  Returns void

enableSmartZoom

- enableSmartZoom(): boolean
  Defined in tsvb.ts:604
  Enable smart-zoom effect.
  Returns boolean

getCustomerId

- getCustomerId(): string

Defined in tsvb.ts:175

Get Customer ID provided by vendor.

Returns string

getEffectProcessor

- getEffectProcessor(): EffectProcessor

  Defined in tsvb.ts:772

  Returns EffectProcessor

getSegmentationPreset

- getSegmentationPreset(): string

  Defined in tsvb.ts:239

  Return current active segmentation mode.

  Returns string

getStream

- getStream(): null | MediaStream

  Defined in tsvb.ts:268

  Get the output MediaStream object for further processing.

  Returns null | MediaStream

hideFps

- hideFps(): boolean

  Defined in tsvb.ts:309

  Hide fps on the stream.

  Returns boolean

run

- run(): boolean

  Defined in tsvb.ts:720

  Run the processing of frames.

  Returns boolean

setAuthRequest

- setAuthRequest(func: ((url: string, payload: Object) => Promise<string>)): void
  - Defined in tsvb.ts:42

  Parameters

  - func: ((url: string, payload: Object) => Promise<string>)

    (url: string, payload: Object): Promise<string>

    Parameters

    - url: string
    - payload: Object

    Returns Promise<string>

  Returns void

setBackground

- setBackground(url: string | MediaStream | MediaStreamTrack | HTMLVideoElement): boolean
  - Defined in tsvb.ts:426

Set media source of the background. Video sources will be played automatically from the beginning.

Parameters

- url: string | MediaStream | MediaStreamTrack | HTMLVideoElement
  the link to image/video of the server or one of the following objects: MediaStream, MediaStreamTrack, HTMLVideoElement.

Returns boolean

setBackgroundColor

- setBackgroundColor(color: number): void
  - Defined in tsvb.ts:260

Set the background color for background 'color' mode.

Parameters

- color: number
  in hexadecimal format.

Returns void

setBackgroundFitMode

- setBackgroundFitMode(mode: string): boolean
  - Defined in tsvb.ts:361

Control background fit/fill mode. Default value is fill mode.

Parameters

- mode: string
  the background fit mode, can be fit of fill

Returns boolean

setBeautificationLevel

- setBeautificationLevel(level: number): boolean
  - Defined in tsvb.ts:387

Control face beautification level.

Parameters

- level: number
  could be from 0 to 1. Higher number -> more visible effect of beautification.

Returns boolean

setBlur

- setBlur(power: number): boolean
  - Defined in tsvb.ts:401

Enable blur of the background and set the power of blur.

Parameters

- power: number
  of the blur, can be a number from 0 to 1. Higher number -> better blur. This value could affect the performance (CPU/GPU, FPS)

Returns boolean

setBoundaryLevel

- setBoundaryLevel(level: number): boolean
  - Defined in tsvb.ts:374

Control segmentation boundaries area.

Parameters

- level: number
  of area, could be from -5 to 5. Higher number -> bigger area outside the segmentation object.

Returns boolean

setBoundaryMode

- setBoundaryMode(mode: string): boolean
  - Defined in tsvb.ts:347

Control boundary mode smooth or strong. Default value is strong mode.

Parameters

- mode: string
  the boundary mode, can be smooth or strong

Returns boolean

setColorCorrectorPeriod

- setColorCorrectorPeriod(value: number): boolean
  - Defined in tsvb.ts:668

Set period in ms for cc-model working.

Parameters

- value: number
  can be a number from 0 to 5000 (default 1000)

Returns boolean

setColorCorrectorPower

- setColorCorrectorPower(value: number): boolean
  - Defined in tsvb.ts:685

Set power of color correction.

Parameters

- value: number
  can be a number from 0 to 1 (default 1)

Returns boolean

setCustomLayout

- setCustomLayout(persent: { size?: number; xOffset?: number; yOffset?: number }): boolean
  - Defined in tsvb.ts:739

Parameters

- persent: { size?: number; xOffset?: number; yOffset?: number }
  - Optional size?: number
  - Optional xOffset?: number
  - Optional yOffset?: number

Returns boolean

setFaceArea

- setFaceArea(value: number): boolean
  - Defined in tsvb.ts:490

Set the face-area proportion. Used by the smart-zoom effect to calculate frame scale value

Parameters

- value: number
  can be a number from 0.01 to 1 (default = 0.1)

Returns boolean

setFaceDetectorAccuracy

- setFaceDetectorAccuracy(value: number): boolean
  - Defined in tsvb.ts:507

Set the face detector accuracy.

Parameters

- value: number
  can be a number from 0.2 to 1 (default 0.75)

Returns boolean

setFilterPart

- setFilterPart(value: number): boolean
  - Defined in tsvb.ts:654

Set filter part for the color correction (dev feature).

Parameters

- value: number
  can be a number from 0 to 1 (default 1)

Returns boolean

setFpsLimit

- setFpsLimit(limit: number): boolean
  - Defined in tsvb.ts:285

Show fps on the stream.

Parameters

- limit: number

Returns boolean

setLayout

- setLayout(mode: string): boolean
  - Defined in tsvb.ts:475

Set the layout mode. Useful for presentations.

Parameters

- mode: string
  could be the one of the following: center, left-bottom, right-bottom

Returns boolean

setSegmentationPreset

- setSegmentationPreset(preset: string): Promise<undefined | true>
  - Defined in tsvb.ts:220

Set the segmentation mode. Segmentation mode allow to choose combination of quality and speed of segmentation. Balanced mode is enabled by default.

Parameters

- preset: string
  in string format. The values could be quality, balanced, speed, lightning.

Returns Promise<undefined | true>

setSmartZoomPerod

- setSmartZoomPerod(value: number): boolean
  - Defined in tsvb.ts:560

Set period in ms for face detector reaction.

Parameters

- value: number
  can be a number from 0 to 1000 (default 100)

Returns boolean

setSmartZoomSensitivity

- setSmartZoomSensitivity(value: number): boolean
  - Defined in tsvb.ts:543

Set sensitivity for the smart-zoom rection. The set value means the difference between the new and old face-params for the smartzoom reaction

Parameters

- value: number
  can be a number from 0 to 1 (default 0.05)

Returns boolean

setSmartZoomSmoothing

- setSmartZoomSmoothing(steps: number): boolean
  - Defined in tsvb.ts:525

Set count of the smart-zoom smoothing. The more steps, the higher the smoothing

Parameters

- steps: number
  can be a number from 0.01 to 1 (default 0.2)

Returns boolean

showFps

- showFps(): boolean
  Defined in tsvb.ts:297

Show fps on the stream.

Returns boolean

stop

- stop(): boolean
  Defined in tsvb.ts:731

Stop the processing of frames.

Returns boolean

switchDrawFaceSquare

- switchDrawFaceSquare(isOn: boolean): boolean
  - Defined in tsvb.ts:577

Switch face-square-drawing mode.

Parameters

- isOn: boolean
  is a boolean argument (default false)

Returns boolean

switchDrawPreFaceSquare

- switchDrawPreFaceSquare(isOn: boolean): boolean
  - Defined in tsvb.ts:592

  Switch preface-square-drawing mode. Draw face square before processing

  Parameters
  - isOn: boolean
    is a boolean argument (default false)

  Returns boolean

toCanvas

- toCanvas(canvas: HTMLCanvasElement): void
  - Defined in tsvb.ts:277

  Set the canvas where will be rendered the processed frames.

  Parameters
  - canvas: HTMLCanvasElement
    the HTMLCanvasElement object.

  Returns void

useStream

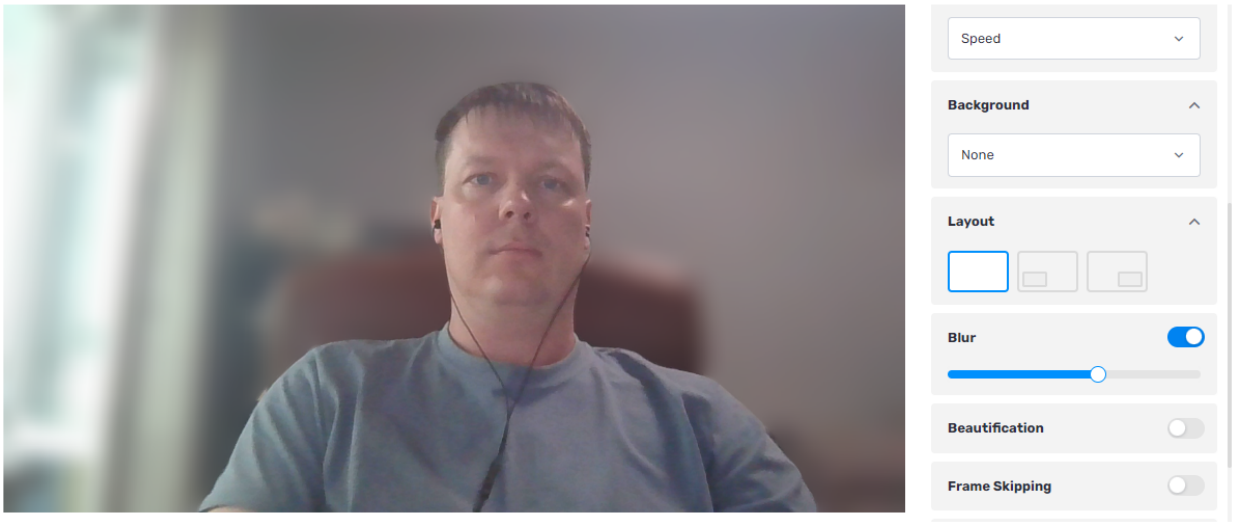
- useStream(stream: MediaStream): void
  - Defined in tsvb.ts:185

  Set the MediaStream object which will be the source of the video frames for processing.

  Parameters
  - stream: MediaStream
    the source MediaStream object.

  Returns void

Демонстрационное приложение для Web



Blur - включение/выключение размытия фона за человеком. Также можно контролировать уровень размытия.

Beautification - включение/выключение сглаживание кожи лица, с возможностью контроля силы применения эффекта.

Background - замена фона за человеком на картинку, видео, однотонный цвет, или захват экрана.

Color Correction - автоматическая коррекция цветов в кадре с возможностью выбора силы применения эффекта.

Smart Zoom - автоматический фрейминг (определение лица в кадре и плавное удержания заданного уровня приближения)

Возможно совместное/параллельное использование всех эффектов.

Camera - выбор камеры.

Resolution - выбор разрешения.

Overlay Screen - заменяет весь видеопоток подготовленным видео или картинкой.

Lower Third - подписи (анимировано показывает заготовленную подпись пользователя)

Segmentation Mode - выбор режима сегментации, чем более качественная сегментация тем больше ресурсов CPU/GPU требуется. Чтобы расширить поддержку слабых устройств мы обучили 4 разные ML модели сегментации (Quality/Balanced/Speed/Lightning).

Frame Skipping - возможность включить пропуск кадров во время сегментации (при включении сегментация будет выполняться на каждый второй кадр). Данный режим также предназначен для слабых пользовательских устройств (снижение нагрузки на CPU/GPU и повышение FPS)