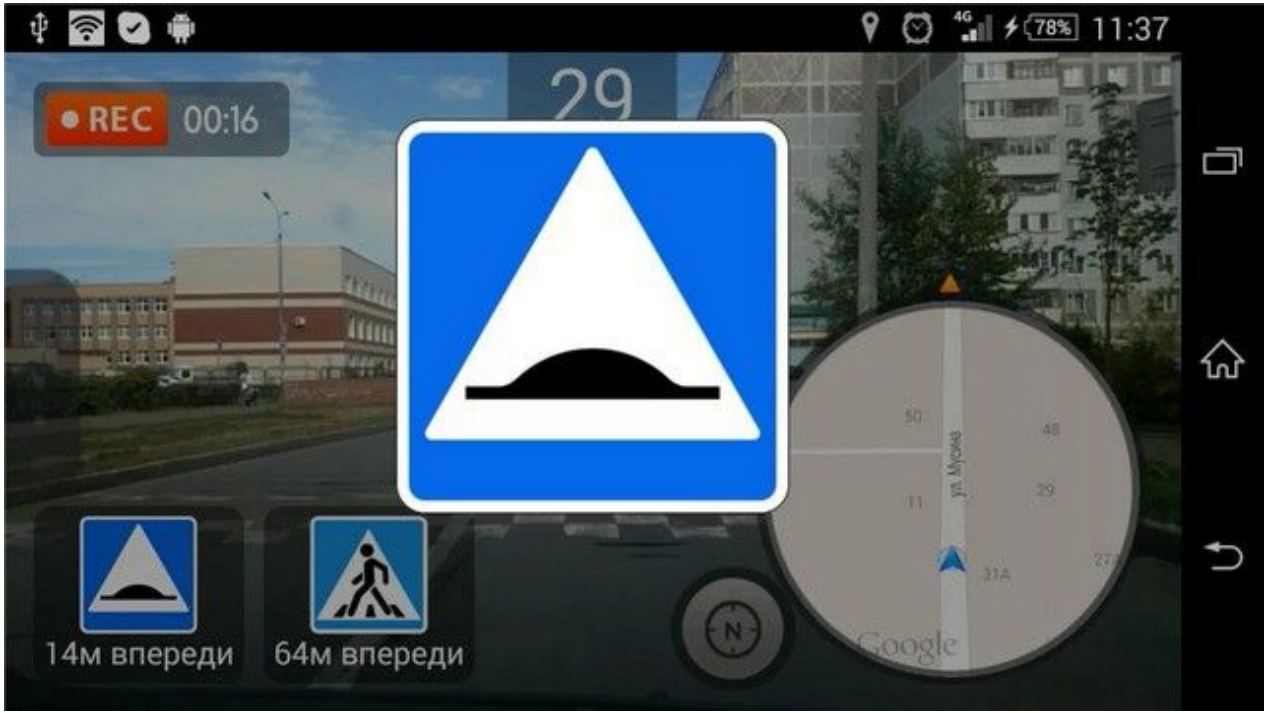
Борьба за FPS и Android-камера, или Как видит зеленый робот

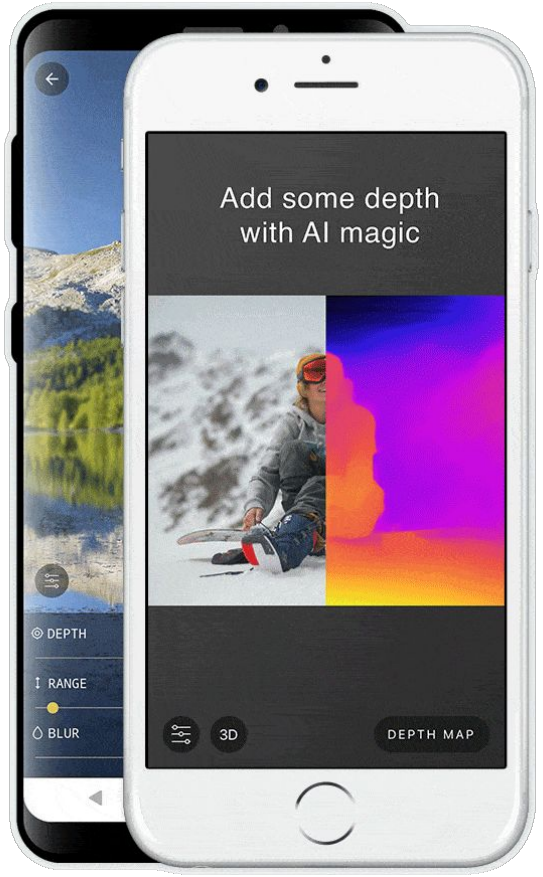
Дмитрий Гордин
@gordinmitya

Кто такой?

Этот Дмитрий Гордин



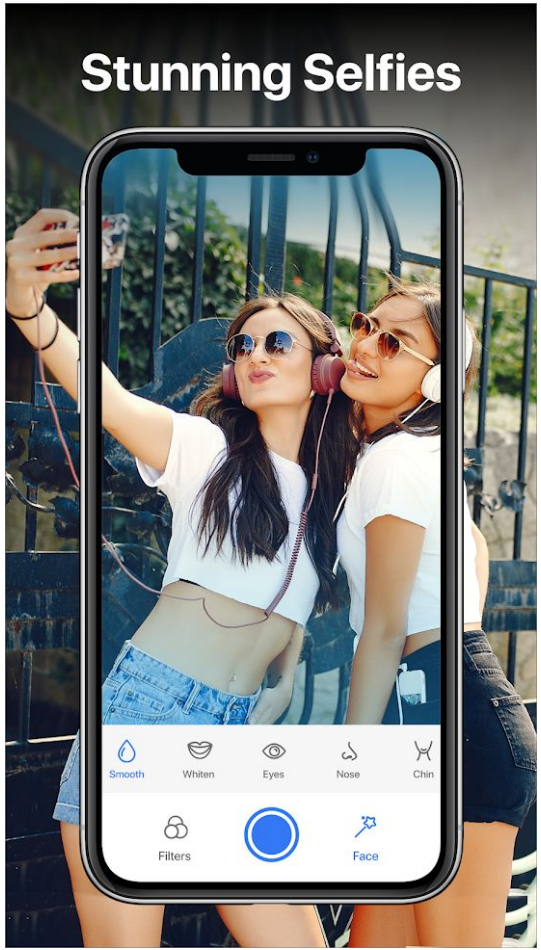
Roadar - умный видеорегистратор

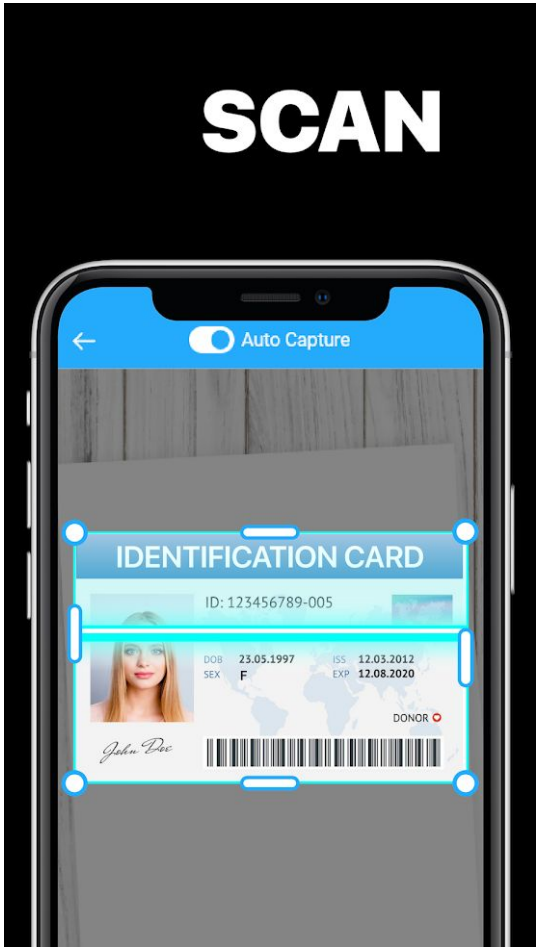


Фоторедактор DPTH.app

Selfix

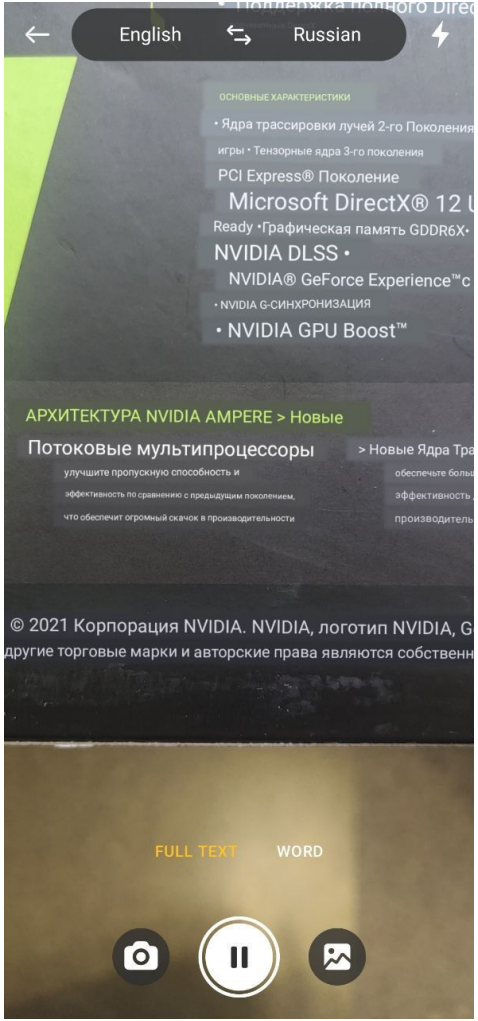
1M+





TapScanner

50M+



Яндекс.Переводчик

Зачем?

Накидаю ключевых слов, чтобы знать, куда гуглить дальше

Поделюсь опытом, на что стоит обратить внимание, а что не сработало

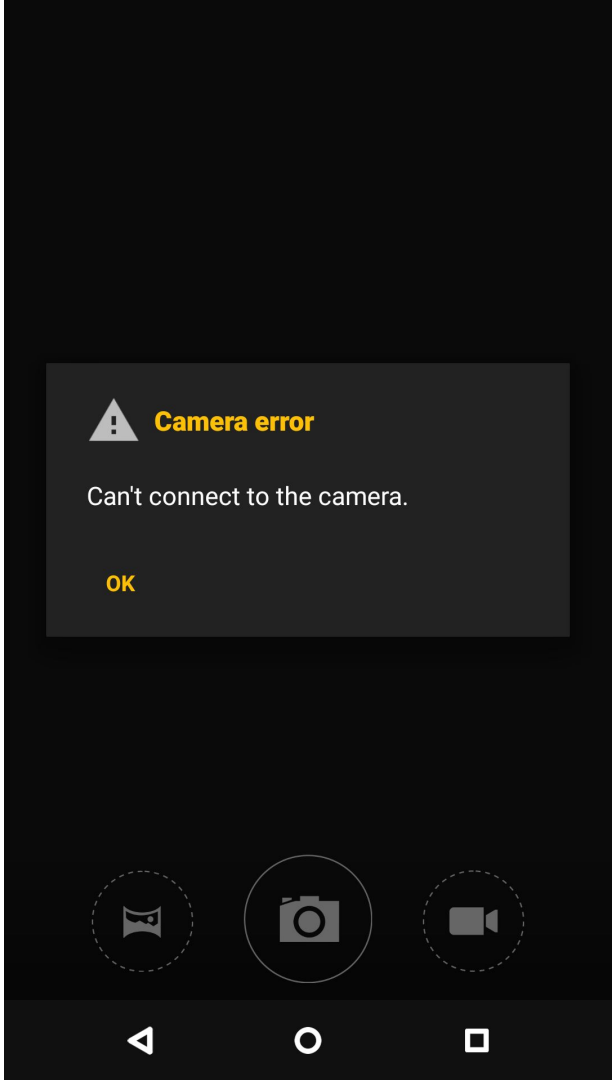
План максимум: заинтересовать, добавить фишечку с камерой в свое приложение

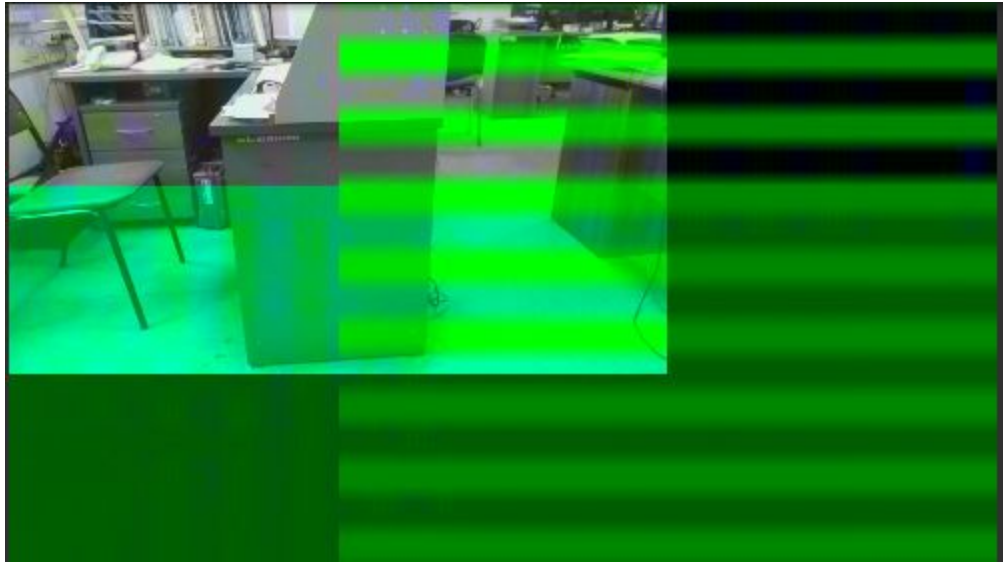
План

- Варнинг;
- 1, 2, X;
- Форматы изображений;
- Обработка картинки;
- Нейросети;
- Отрисовка результата.

а пока вопрос - у кого из присутствующих в приложении есть фичи с камерой?

Предупреждение


Camera error
Can't connect to the camera.
OK



Нужно снять фото или видео

Intent ACTION_IMAGE_CAPTURE

+ пользователь с ней уже знаком, привык;
+ встроенные фишки от вендора телефона;
+ наиболее протестированный вариант.

как, например, Яндекс.Карты делают для отзывов

Нужно снять фото или видео

какая-нибудь либа с github, CameraKit 5.1k ⭐

+ не покидая приложения;
+ возможность добавить кнопочки/цвета по дизайну;
+ есть кому пожаловаться;
+ из коробки может быть обработка, кропалки/крутилки.

Нужно снять фото или видео

CameraX

+ Нууу очень кастомная вьюшка;
+ ???

- Проведи время с семьей;
- Займись спортом;
- Выплати тех долг, порефактори легаси;
- Изучи новый язык.

*СКРИНШОТ **ВАШЕГО** ПРИЛОЖЕНИЯ СО СЛОМАННОЙ КАМЕРОЙ*

Не делай так

Camera Api 1 или 2

- приходят креши, а у вас действительно "все работает на моем устройстве";
- еще хуже, если креши **не** приходят, а функционал просто тихонько отвалился.

*История как я **не** воспользовался каршерингом*

Стартуем!

Camera 1

+ Гораздо проще и понятнее API;
+ Стабильнее;
+ Отдает byte[] с понятным NV21;
+ Telegram использует Api 1, недавно в контесте было задание затащить вторую, но не затащили.

- Не отдает напрямую в OpenGL.

Совет

---

addCallbackBuffer

Подсовывай несколько буферов заранее;

Переиспользуй буферы.

Camera 2

- Может все и ничего;
- На бюджетных телефонах это обертка над Camera 1;
- Сложное, запутанное, непонятное;
- Бессмысленное и беспощадное.

+ Может рисовать в OpenGL текстуру.

Camera 2

OpenCamera на sourceforge.net;

Посмотреть что-нибудь по теме можно в WebRTC;

Telegram использует для видеозвонков.

Стейт-машина

```
sealed class State {
    object IDLE : State() {
        fun open(params: OpenRequestParams) = OPENING(params, attempt: 0)
        fun loop() = this
        override fun toString(): String = "IDLE"
    }

    object CLOSING : State() {...}

    class CLOSING_TO_REOPEN internal constructor(val params: OpenRequestParams) : State() {...}

    class OPENING internal constructor(val params: OpenRequestParams, val attempt: Int) : State()

    class WORKING internal constructor(val session: CameraCaptureSession) : State() {...}
```

Camera X

Еще одно Api точно решит проблему, много апи не бывает;

Обертка над Camera 2;

Иногда, изредка, что-то умеет на некоторых телефонах;

Активно пилится, есть надежда.

CameraX – bokeh effect

Получаем кадр

ImageReader

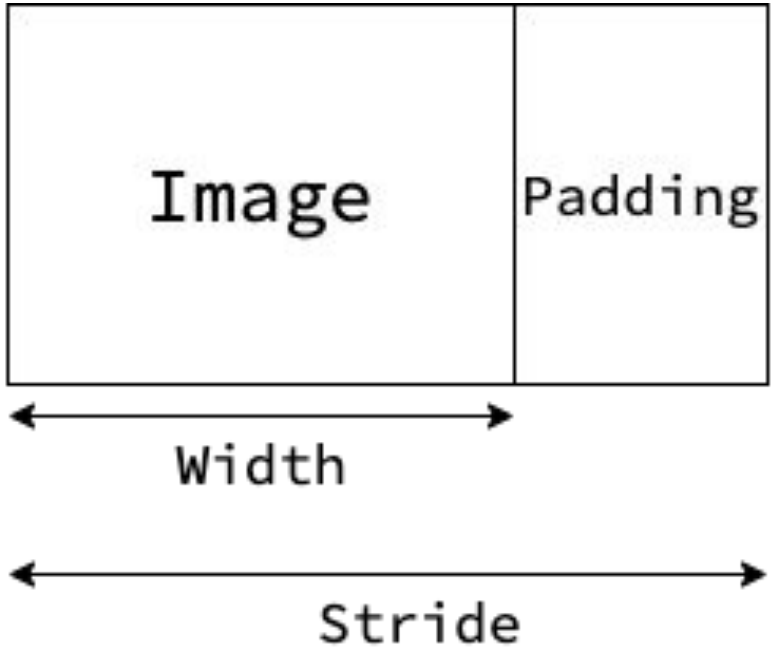
Читает из всего, что может писать в Surface:

- Camera 2 / X;
- MediaCodec, MediaPlayer.

Отдает в неудобном виде – массив Image.Plane

Image.Plane

- ByteBuffer
- pixelStride
- **rowStride**



Yuv2Buf

Преобразуем 3x Plane в нормальный ByteBuffer

github.com/gordinmitya/**yuv2buf**

Эт мое (:

Вмержено в android/camera-samples, 347 строчек

События состояния камеры

Camera1:

AutoFocusMoveCallback - true/false

Camera2, CameraX:

CONTROL_AF_STATE - фокусировка

CONTROL_AE_STATE - экспозиция

CONTROL_AWB_STATE - баланс белого

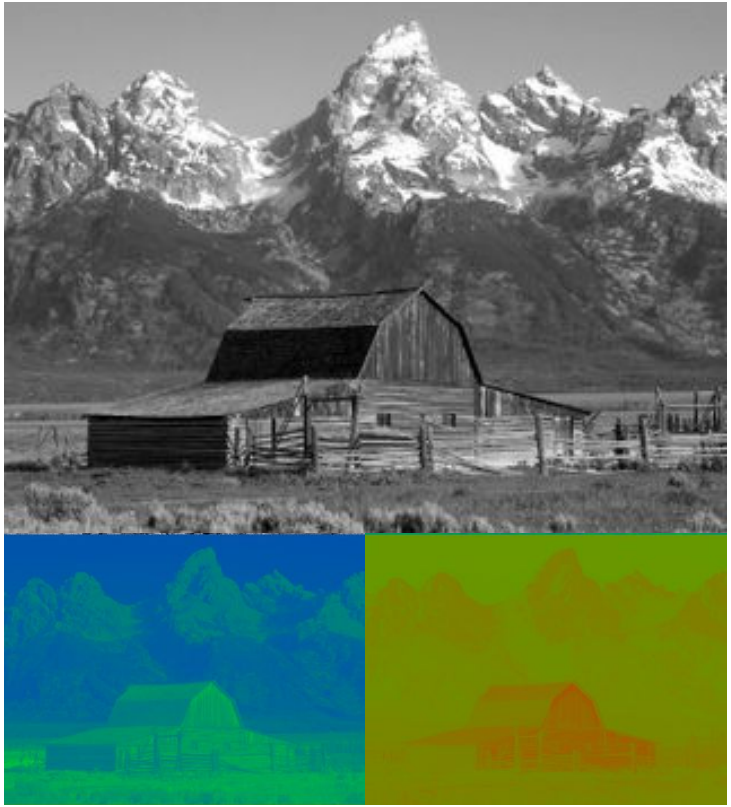
Камеры

Camera1 - byte[], NV21;

Camera2 - ImageReader, yuv2buf, NV21 (почти всегда);

CameraX == Camera2.

А как работаете с камерой вы?



YUV

Single Frame YUV420 NV21:

Y:

PixelStride = 1

U == V:

PixelStride = 2

Position in byte stream:

Blackview BV6000S
Android 7.0

Single Frame YUV420:

Y:

PixelStride = 1

U == V:

PixelStride = 1

Position in byte stream:

Blackview BV6000S
Android 7.0

И что теперь с этим делать?

CPU

- ★ Qualcomm Snapdragon
- ★ HiSilicon Kirin
- ★ Samsung Exynos
- ★ Mediatek Helio
- ★ Intel Atom

Многопоточность

Single **I**nstruction **M**ultiple **D**ata, NEON

Гораздо быстрее памяти, есть кеши, всего несколько ядер.

Работать с YUV

- Использовать только Y канал
  - Черно-белое изображение
  - Трекинг
  - QR / Barcode

- Переучить сеточку принимать yuv на вход
  - Сжимать Y до размеров u и v

Конвертировать в RGB

- Java, C++ - слишком долго *~50ms;*
- RenderScript Intrinsics *~2.2ms* (deprecated);
- OpenCV *~1ms;*
- Встроенные в фреймворк тулзы;
- OpenGL.

Чем обрабатывать?

---

java - на удивление можно

github.com/zxing/zxing - детекция qr кодов

OpenCV

C++ и OpenCV

- весит приемлемо ~N MB;
- за счет NEON очень быстрый;
- Кроссплатформенно
  - Тестируешь на компьютере,
  - Запускаешь на Android и iOS.

GPU

- ★ Adreno
- ★ Mali

- ★ OpenGL + Compute Shaders
- ★ Vulkan
- ★ OpenCL
- ★ RenderScript

Очень слабые ядра по одиночке, но зато ядер очень много.

RenderScript – deprecated с Android 12

Платформа сама определяет, на чем запустить CPU или GPU

Только под Android

В случае с DPTH и хитрой функцией blur - оказалось гораздо быстрее, чем OpenGL

Есть эффективные встроенные операции

OpenGL ES

Легко ошибится и сложно отлаживать

Есть готовый набор шейдеров GpuImage

С версии 3.1 (android 5+) появились Compute Shaders

Какая версия на телефоне у пользователя – не гарантируется

Как работать в Android – github.com/google/**grafika**

Должен быть быстрее OpenGL;

Удобнее в разработке чем OpenGL;

Заточен именно под вычисления;

Используется в tflite, MNN, TNN;

Поддержка на телефоне не гарантируется.

Vulkan

Кажется, сложнее чем OpenGL;

Советуют переходить с RenderScript.

Нейросети

NPU

★ DSP - Qualcomm
★ Kirin NPU - HiSilicon
★ Exynos NPU - Samsung
★ APU - MediaTek

У каждого свое апи

\+ NNAPI от Android

Очень разные API, поддерживают разный набор операций.

Frameworks

- Tensorflow Lite
- TNN by Tencent
- MNN by Alibaba
- OpenCV DNN
- MACE by Xiaomi
- SNPE by Qualcomm
- ncnn by Tencent
- Tensorflow Mobile
- Paddle-Lite by Baidu
- HiAI by Huawei
- PyTorch Mobile
- Samsung Neural SDK
- NNPACK
- Tengine by ARM
- ML-Kit
- …

ai-benchmark

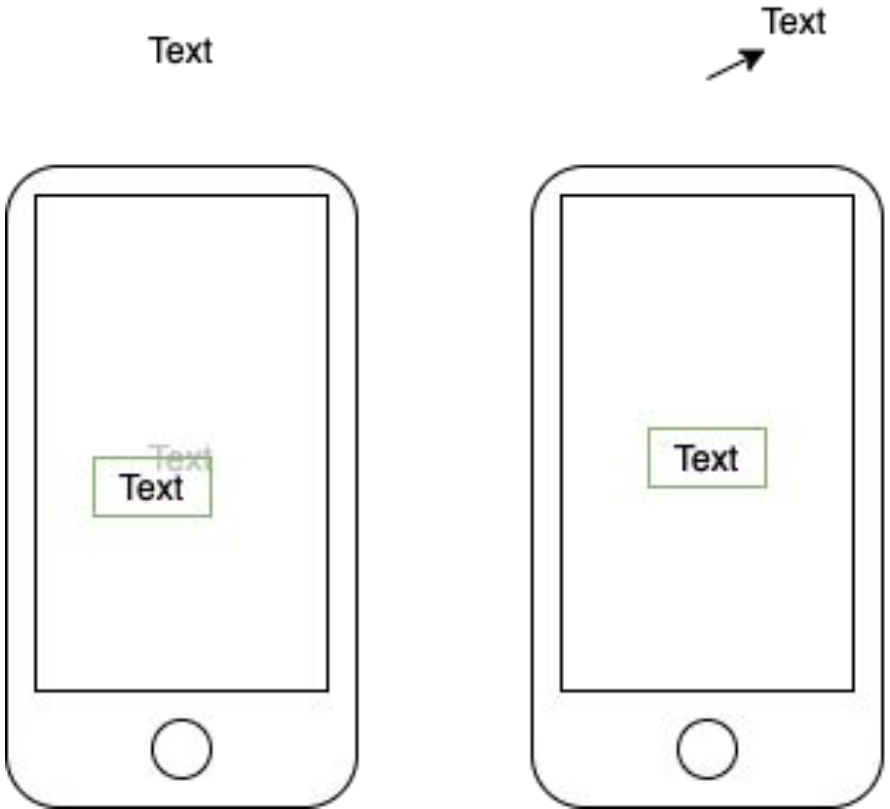
Model	SoC	RAM	Year	Android	Updated	Lib
Huawei P50 Pro	Kirin 9000	8GB	2021	10	7.21	nn
Huawei Mate 40 Pro	Kirin 9000	8GB	2020	10	6.21	nn
Huawei Mate 40 Pro+	Kirin 9000	12GB	2020	10	6.21	nn
Oppo Find X3 Pro	Snapdragon 888	12GB	2021	11	6.21	nn
LG V70	Snapdragon 888	8GB	2021	11	4.21	nn
Sony Xperia 1 III	Snapdragon 888	12GB	2021	11	7.21	nn
Samsung Galaxy S21 Ultra	Exynos 2100	12GB	2021	11	4.21	ne
Samsung Galaxy S21 Ultra	Snapdragon 888	12GB	2021	11	6.21	nn
Samsung Galaxy S21+	Snapdragon 888	8GB	2021	11	6.21	nn
Samsung Galaxy S21	Snapdragon 888	8GB	2021	11	6.21	nn
OnePlus Nord 2 5G	MediaTek Dimensity 1200-AI	8GB	2021	11	6.21	mm
Samsung Galaxy S21 Ultra	Exynos 2100	12GB	2021	11	2.21	nn
Huawei Mate 40	Kirin 9000E	8GB	2020	10	7.21	nn
Samsung Galaxy S21+	Exynos 2100	8GB	2021	11	2.21	nn
Samsung Galaxy S21	Exynos 2100	8GB	2021	11	2.21	nn
Asus ROG Phone 5	Snapdragon 888	16GB	2021	11	6.21	nn
Zenfone 8	Snapdragon 888	16GB	2021	11	6.21	nn

Profiling

```
Sort by time cost !
Node Type       Avg(ms)         %               Called times    Flops Rate
Eltwise         0.096333        0.218235        4.000000        0.072375
ReLU            0.406333        0.920514        19.000000       0.391825
Interp          0.450000        1.019437        1.000000        1.277800
Crop            0.461667        1.045867        8.000000        0.219622
BinaryOp        0.786333        1.781372        23.000000       0.501636
Pooling         1.553667        3.519701        1.000000        0.159725
ConvertTensor   1.575001        3.568031        51.000000       1.115579
Deconvolution   5.749334        13.024632       4.000000        18.088249
Convolution     33.063358       74.902252       75.000000       78.173134
total time : 44.142010 ms, total mflops : 313.038055
main, 112, cost time: 139.851013 ms
```

Пора рисовать

Задержка в отрисовке vs задержка камеры

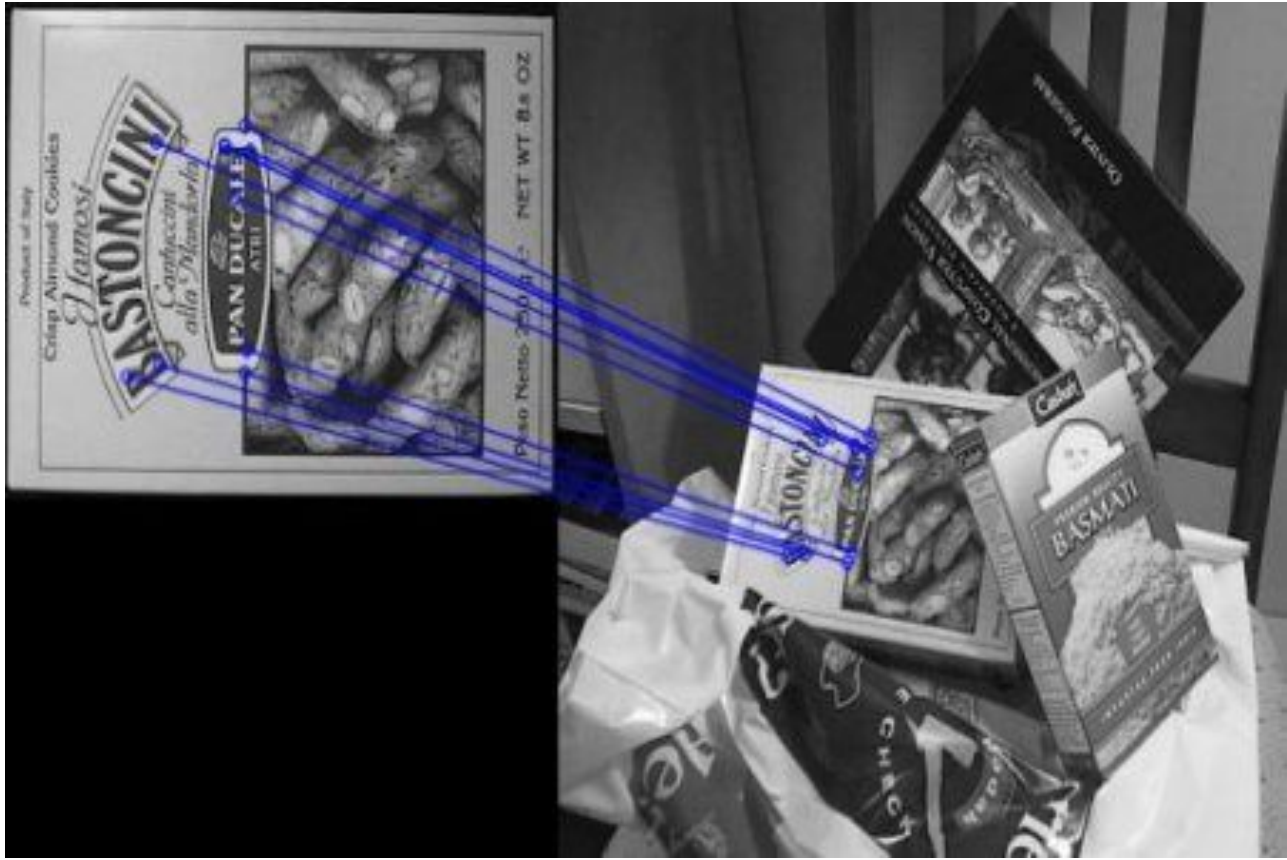


Трекинг

KLT

Feature matching

homography matrix



Контакты

Дмитрий Гордин

Заходите обсудить:

t.me/deepmobile

@gordinmitya tw,tg,github

У нас есть вакансии!

Яндекс

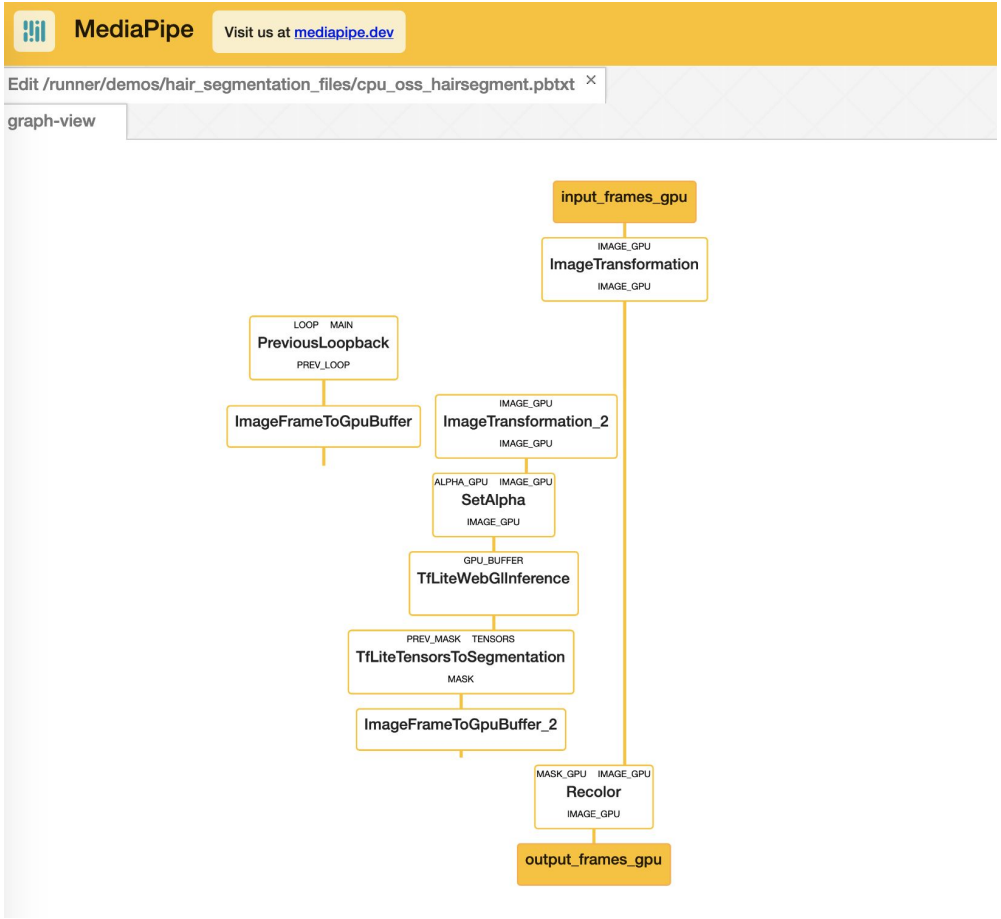
Android & iOS

Пишите за рефералочкой ⊂(◉‿◉)⊃

Материалы

Mediapipe

Для сложных сценариев



Работа с OpenGL

---

github.com/google/grafika

github.com/cats-oss/android-gpuimage

github.com/wysaid/android-gpuimage-plus

RenderScript / Vulkan

---

developer.android.com/guide/topics/renderscript/migrate

Camera2

---

sourceforge.net/projects/opencamera

chromium.googlesource.com/.../webrtc/CameraEnumerationAndroid.java

webrtc.googlesource.com/src/+/refs/heads/main/sdk/android/api/org/webrtc/

github.com/gordinmitya/yuv2buf

Фреймворки для нейросетей

---

github.com/alibaba/MNN

github.com/tencent/tnn

ai-benchmark.com/ranking.html

Доклады

Выступление Ben Sandofsky — Building a Realtime video processor with Swift and Metal

Андрей Володин - Байтик к байтику, или как выжать из телефона всё и не расплавить его