




ОНЛАЙН-ОБРАЗОВАНИЕ

# Онлайн-образование





# Меня хорошо видно && слышно?

Ставьте  , если все хорошо  
Напишите в чат, если есть проблемы



# Защита проекта

## Тема: «Система детектирования лиц»

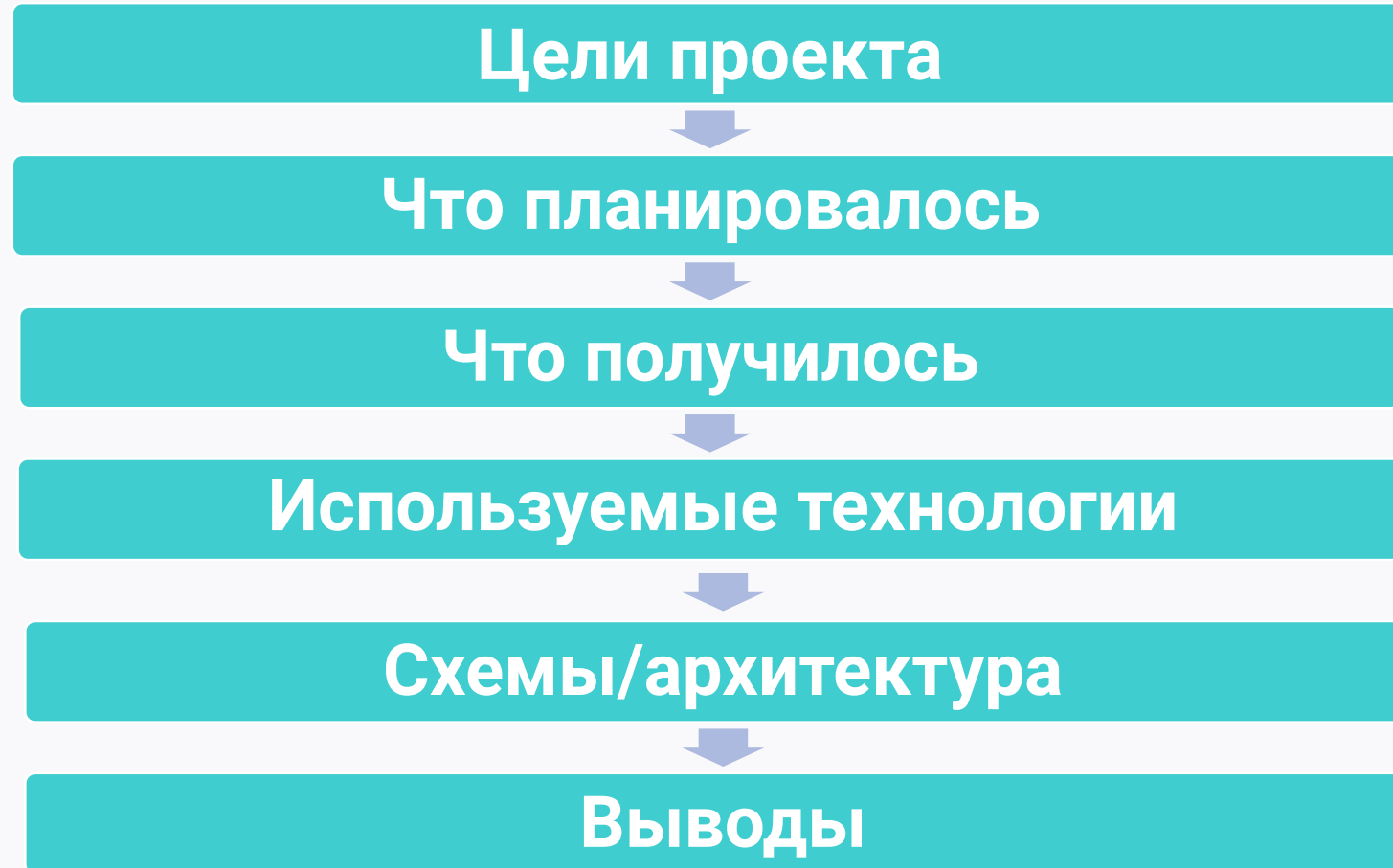


Kseniia Terekhova

Computer Vision Enthusiast



# План защиты



# Цели проекта

1 Использование C++ версий библиотек компьютерного зрения

2 Работа с предобученными моделями с точки зрения Software Engineer

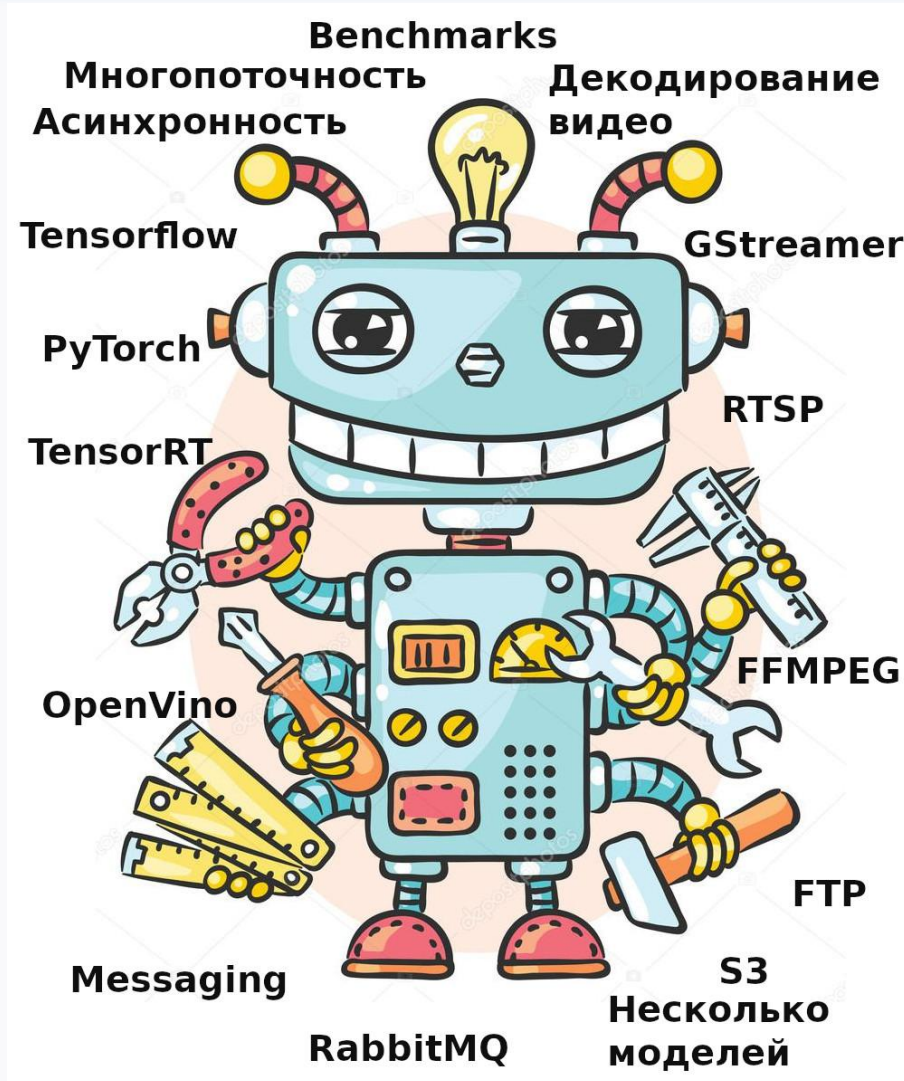
3 Многопоточная и асинхронная обработка изображений и стримминг видео

# C++ в компьютерном зрении

- 1 Многие библиотеки компьютерного зрения имеют C++ API наряду с Python
- 2 C++ - язык для performance critical приложений (на фоне оптимизаций нейросетей под CPU, GPU, TPU, VPU, NPU и т. п.)
- 3 Непосредственное управление памятью (на фоне работы с большими изображениями и видео)
- 4 Язык для системного программирования, работающий “близко к устройству” (на фоне широкого спектра устройств, пригодных для запуска нейросетей)
- 5 При наличии предобученных моделей под многие стандартные задачи, остается достаточное количество инженерной работы



# Что планировалось



Выделение  
игурой/маркеры  
м инфы

ание  
роса



# Что планировалось

1 Использование готовой оптимизированной модели на GPU

2 Задача - детектирование лиц

3 Стримминг видео в веб-интерфейс

4 Многопоточность, асинхронность и параллельность

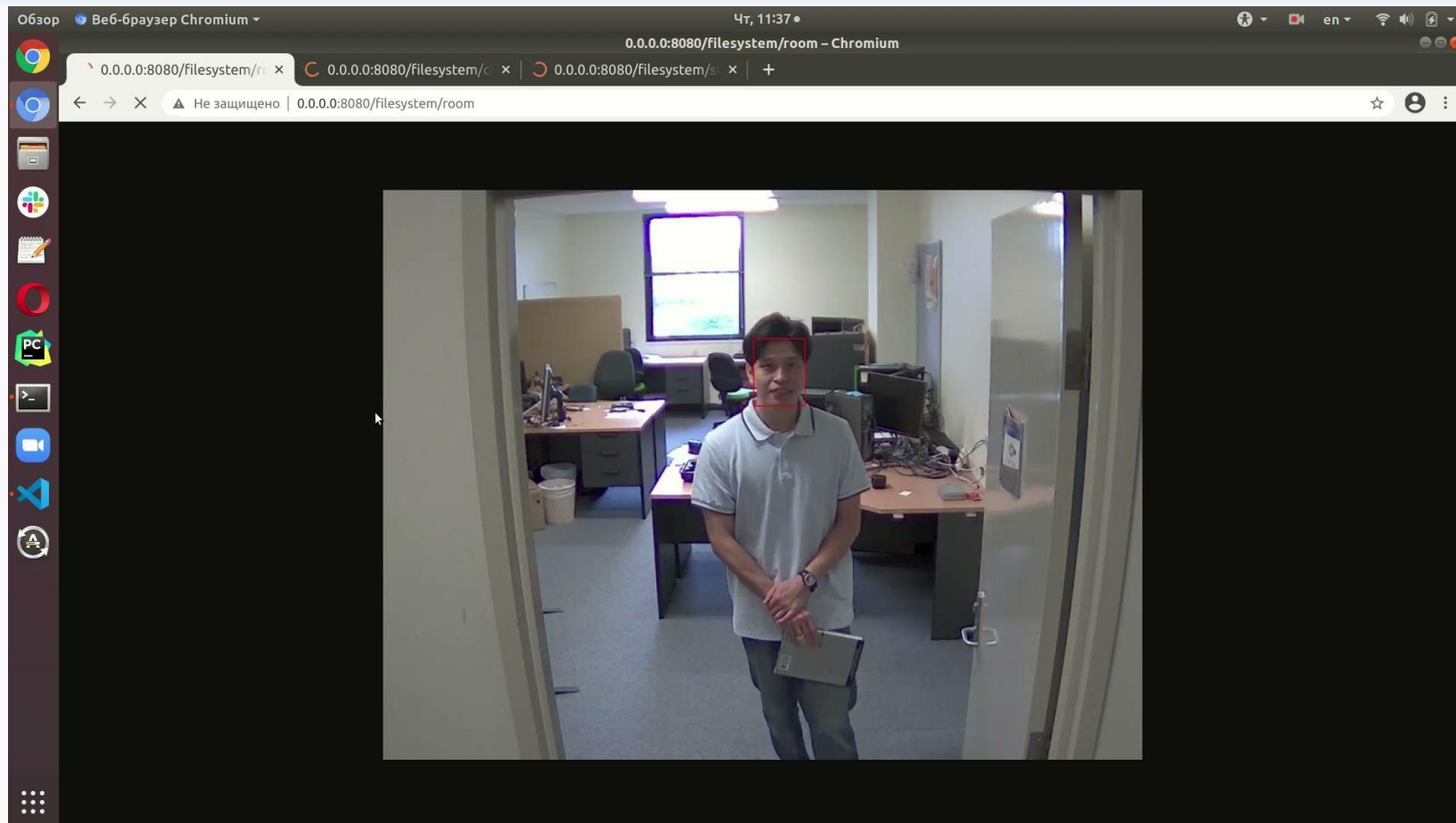
5 Данные на диске с возможностью добавления альтернативных источников

ение  
маркеры





# Что получилось



<https://github.com/prickly-u/inference-server/tree/main/inference/ultraFaceOnnx>

# Используемые технологии

1 TensorRT

2 Ultraface ONNX

3 Motion JPEG

4 Boost beast/boost asio

5 OpenCV

Выделение  
группы/маркеры  
м инфы

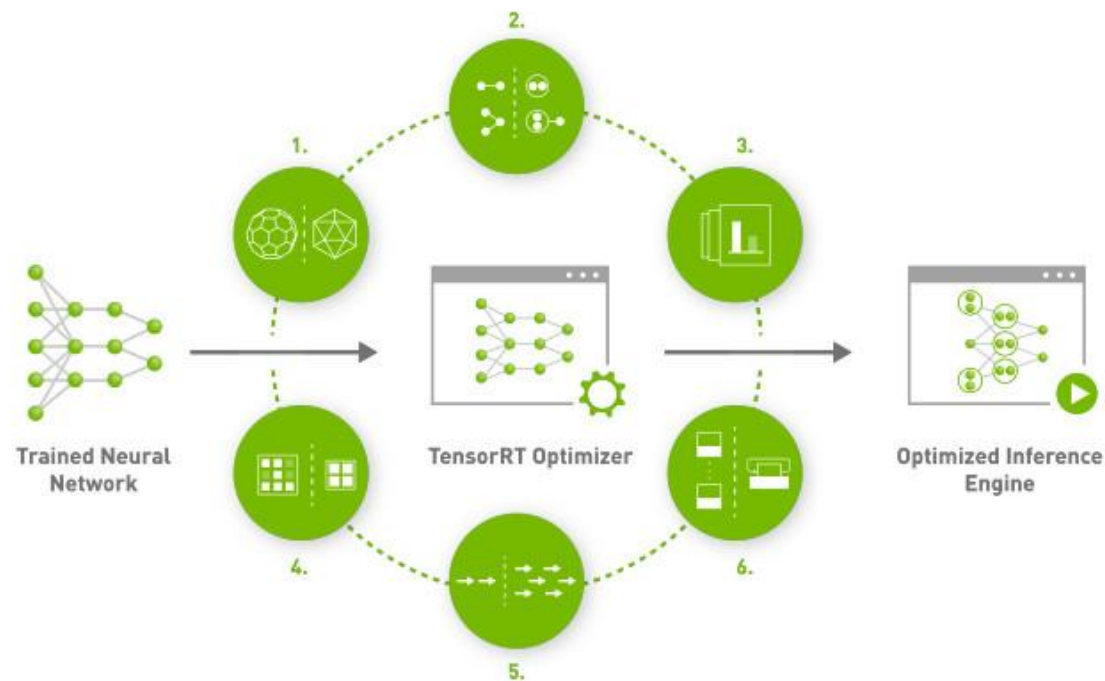
ние  
оса





# TensorRT

NVIDIA TensorRT is an SDK for high-performance deep learning inference. It includes a deep learning inference optimizer and runtime that delivers low latency and high throughput for deep learning inference applications.



# Поддержка форматов в TensorRT

ONNX - открытый формат моделей машинного обучения, позволяющий переносить их между разными фреймворками и инструментами

Caffe

UFF (for Tensorflow)



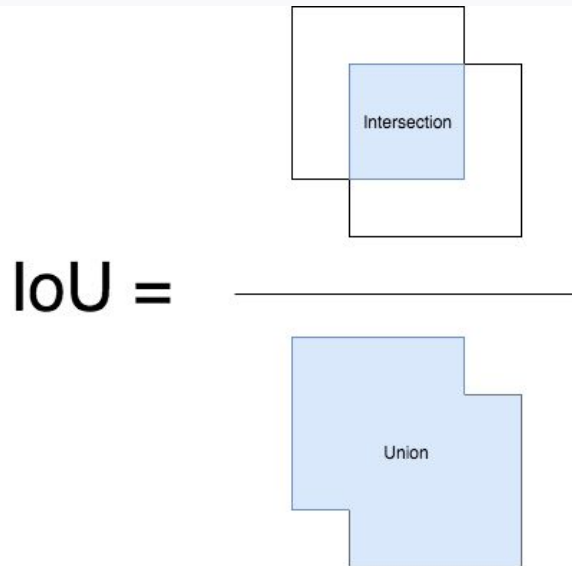
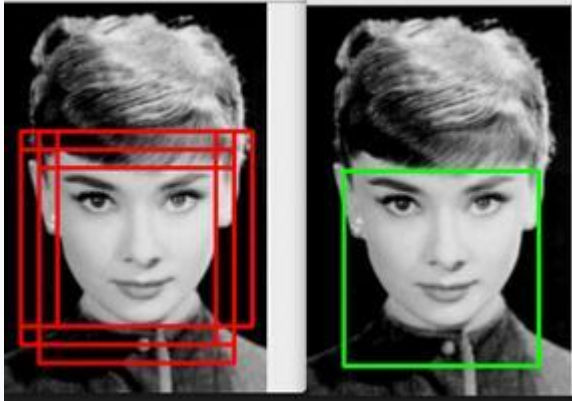


# UltraFace ONNX

- Легковесная модель для детектирования лиц на периферийных устройствах
- ONNX-формат
- Входной тензор: "input",  
batch\_size x 3 channels x 320 width x 240 height
- Формат каналов: BGR
- Препроцессинг:  $(\text{pixel} - 127.0) / 128.0$
- Выходные тензоры:
  - scores: 4420 x 2 classes (face/background)
  - boxes: 4420 x 4 coordinates



# Non-maximum suppression



## Algorithm 1 Non-Max Suppression

```
1: procedure NMS( $B, c$ )
2:    $B_{nms} \leftarrow \emptyset$  Initialize empty set
3:   for  $b_i \in B$  do => Iterate over all the boxes
4:      $discard \leftarrow \text{False}$  Take boolean variable and set it as false. This variable indicates whether b(i) should be kept or discarded
5:     for  $b_j \in B$  do Start another loop to compare with b(i)
6:       if  $\text{same}(b_i, b_j) > \lambda_{nms}$  then If both boxes having same IOU
7:         if  $\text{score}(c, b_j) > \text{score}(c, b_i)$  then
8:            $discard \leftarrow \text{True}$  Compare the scores. If score of b(i) is less than that of b(j), b(i) should be discarded, so set the flag to True.
9:         if not  $discard$  then Once b(i) is compared with all other boxes and still the discarded flag is False, then b(i) should be considered. So add it to the final list.
10:           $B_{nms} \leftarrow B_{nms} \cup b_i$ 
11:   return  $B_{nms}$  Do the same procedure for remaining boxes and return the final list
```





# Motion JPEG over HTTP

Стриминг видео по HTTP

```
Content-Type: multipart/x-mixed-replace; boundary="frame"
```

```
--frame
```

```
Content-Type: image/jpeg
```

```
<jpeg bytes>
```

```
--frame
```

```
Content-Type: image/jpeg
```

```
<jpeg bytes>
```

```
... and so on ...
```

```
--frame--
```



# Альтернативы Motion JPEG

RTPS-поток

GStreamer

FFMPEG





# Boost beast/Boost asio

Многопоточный http-сервер

Асинхронные обработчики событий

Стриминг видео в M-JPEG

Показ кадров по steady\_timer

Strands для обработки без блокировок

Выделение  
группы/маркеро  
м инфы

ние  
оса



# OpenCV

`cv::imread`

`cv::resize`

`cv::rectangle`

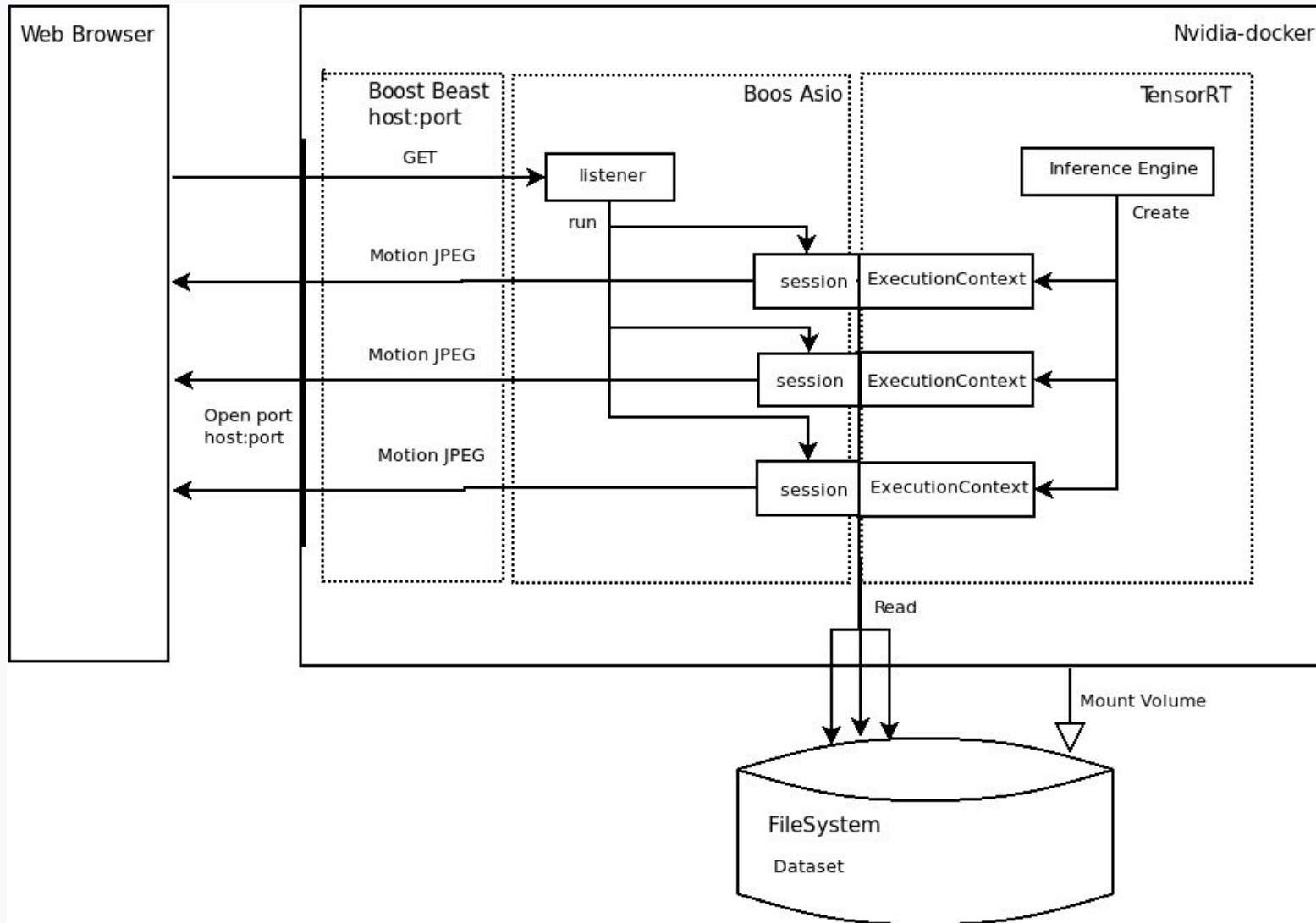
`cv::imencode`

`cv::Mat, cv::Point, cv::Scalar`

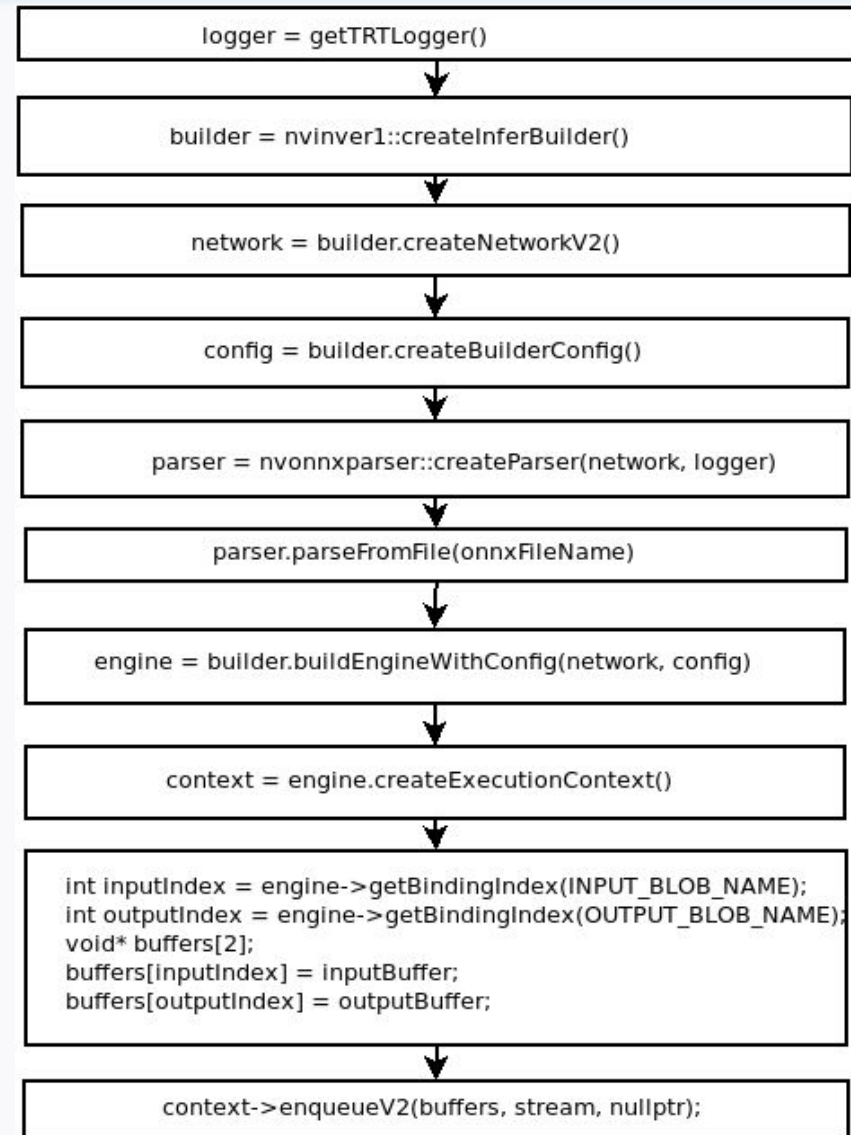




# Верхнеуровневая архитектура



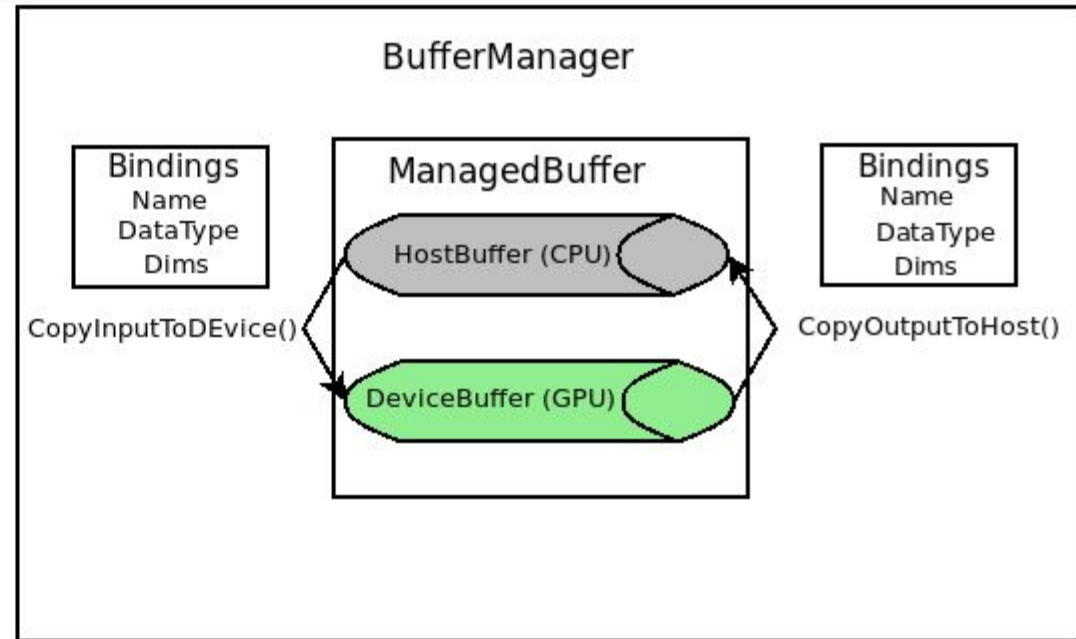
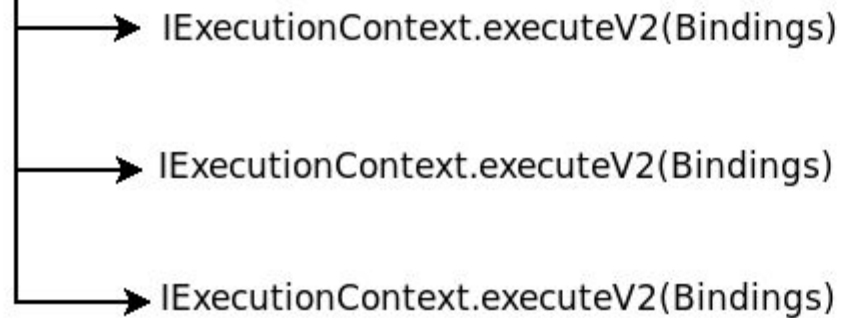
# Инференс модели на TensorRT



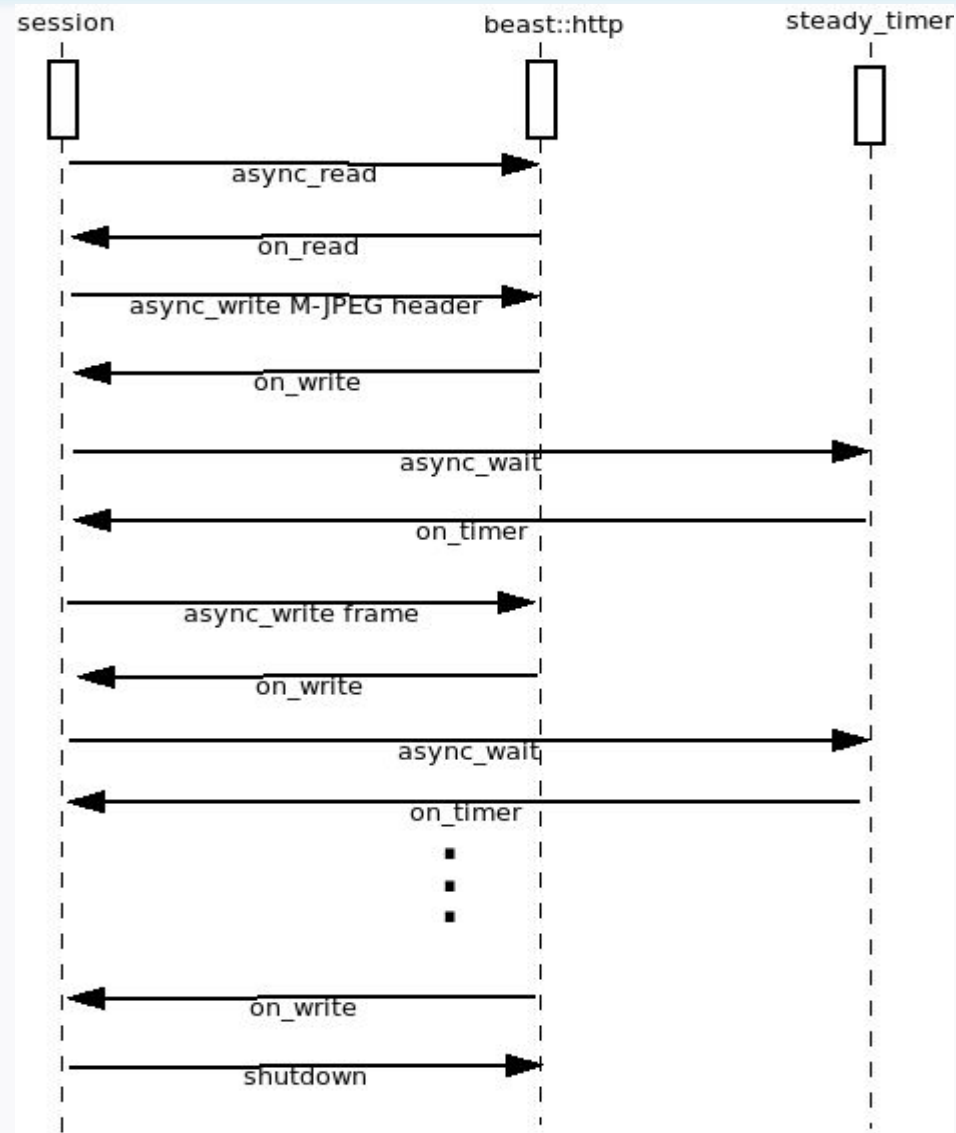


# Параллельное выполнение контекстов

ICudaEngine.createExecutionContext()

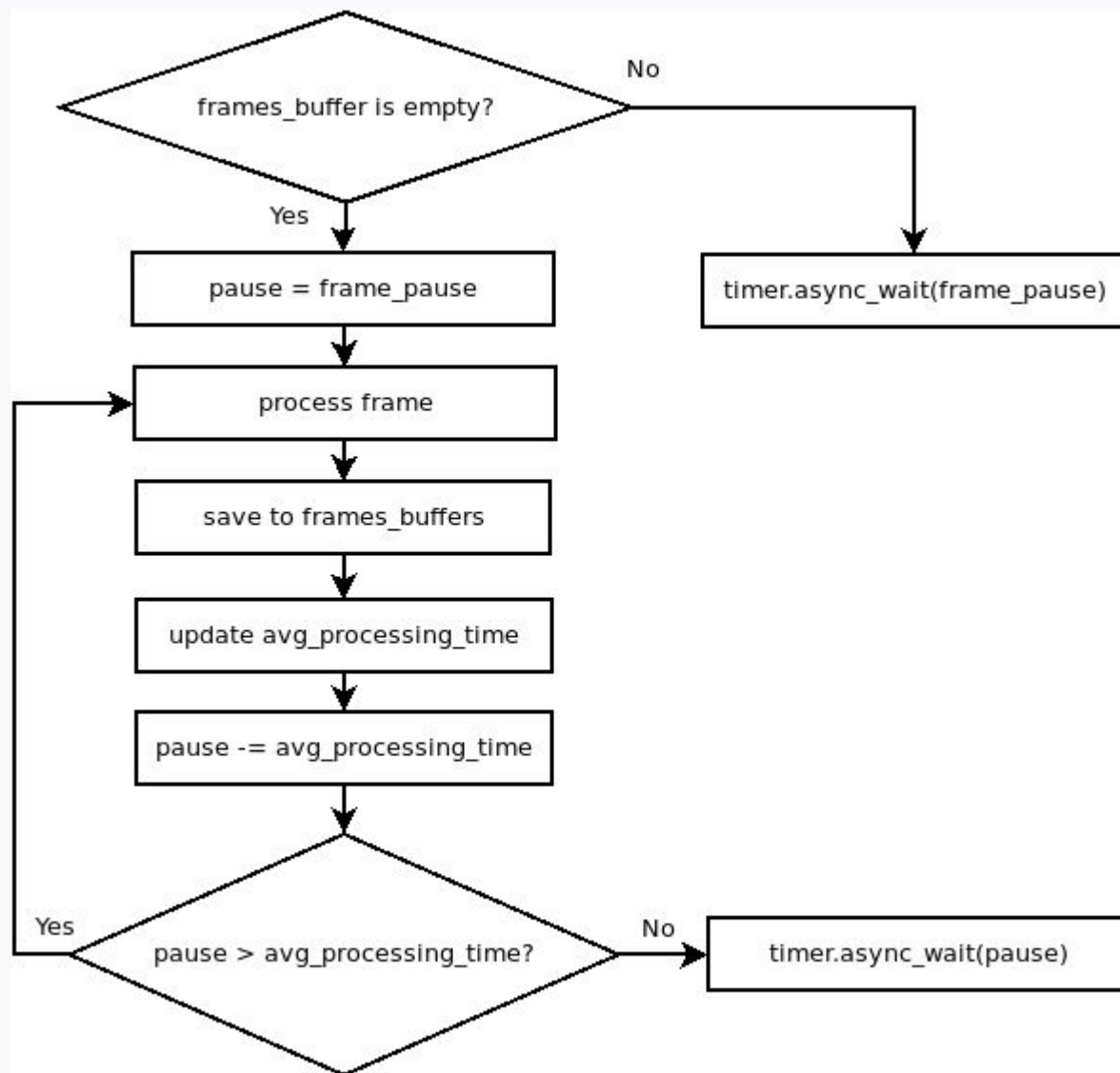


# Async handlers



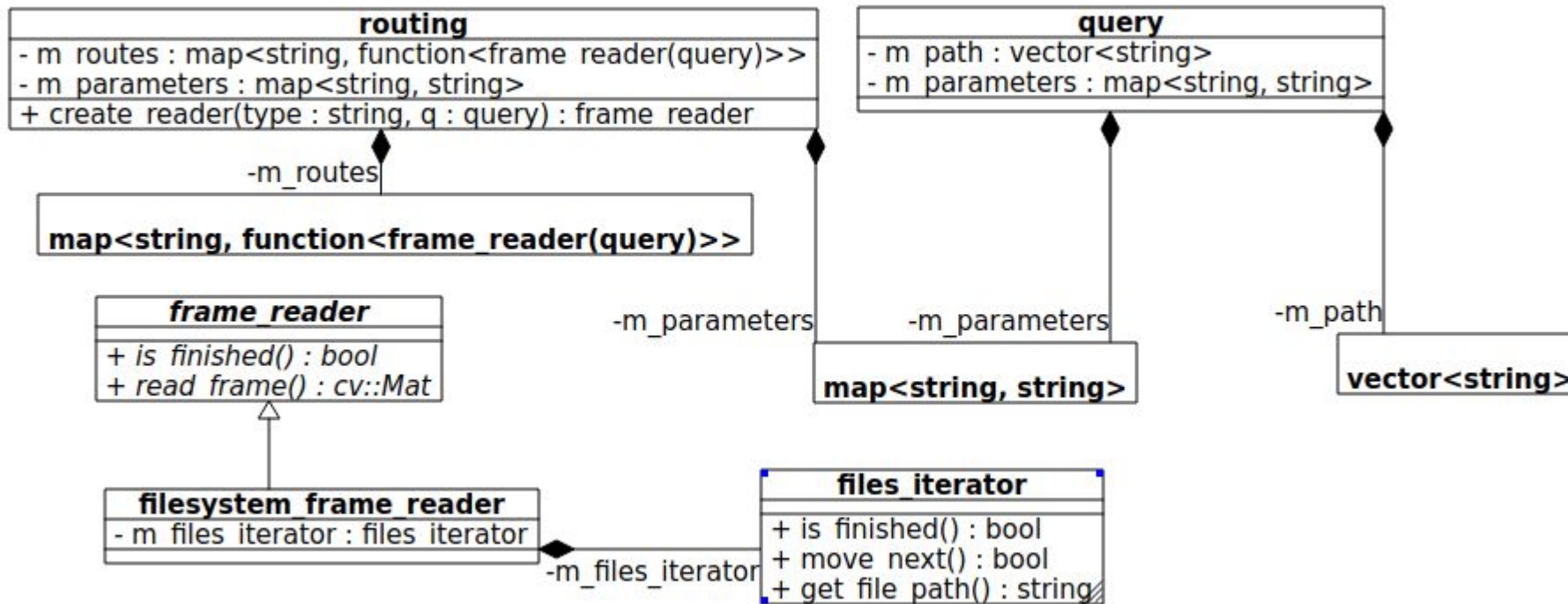


# Подготовка кадров



# Простейший routing в стиле REST

http://0.0.0.0:8080/filesystem/corridor?ext=jpg



# Выводы и планы по развитию

## Выводы:

- Компьютерное зрение достигло качества и доступности, достаточных для решения многих прикладных задач;
- На данный момент в области есть не только исследовательская, но и инженерная работа;

## Возможные пути развития приложения:

- Замеры производительности, оптимизация, автоматическая адаптация под мощности сервера;
- Использование современных подходов к работе с видео (RTSP, GStreamer, FFMPEG и т.п.)
- Использование различных источников данных помимо диска (FTP, Amazon S3, Messaging и т.п.)





The background of the slide features an aerial view of a city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer that contains a white network pattern of dots and lines. The text "Спасибо за внимание!" is centered in this blue area.

Спасибо за внимание!