人脸识别应用实战：SeetaFace2

- 李凯周@SeetaTech
- 2019年09月17日

Outlines

- SeetaFace2简介
- SeetaFace2接口介绍
  - 检测、定位、识别
- SeetaFace2应用举例
  - 跟踪、质量评估
  - 人证对比
  - 动态识别
- SeetaFace2源码分析
  - SeetaNet
  - 前处理与后处理
- 职业分享：AI工程师

人脸认证

人脸闸机

动态识别

人脸检索

人脸门禁

人证对比

- **SeetaFace2** 采用标准 C++ 开发，全部模块均不依赖任何第三方库，支持 x86 架构（Windows、Linux）和 ARM 架构（Android）。SeetaFace2 支持的上层应用包括但不限于人脸门禁、无感考勤、人脸比对等。

- SeetaFace2: https://github.com/seetafaceengine/SeetaFace2

SeetaFace2接口介绍

SeetaTech
中科视拓

- FaceDetector 人脸检测
- FaceLandmarker 人脸关键点定位
- FaceRecognizer 人脸识别

FaceDetector

包含头文件：

```
#include <seeta/FaceDetector.h>
#include <seeta/Struct_cv.h>
```

检测结果：

```
struct SeetaFaceInfo
{
    SeetaRect pos;
    float score;
};

struct SeetaFaceInfoArray
{
    struct SeetaFaceInfo *data;
    int size;
};
```

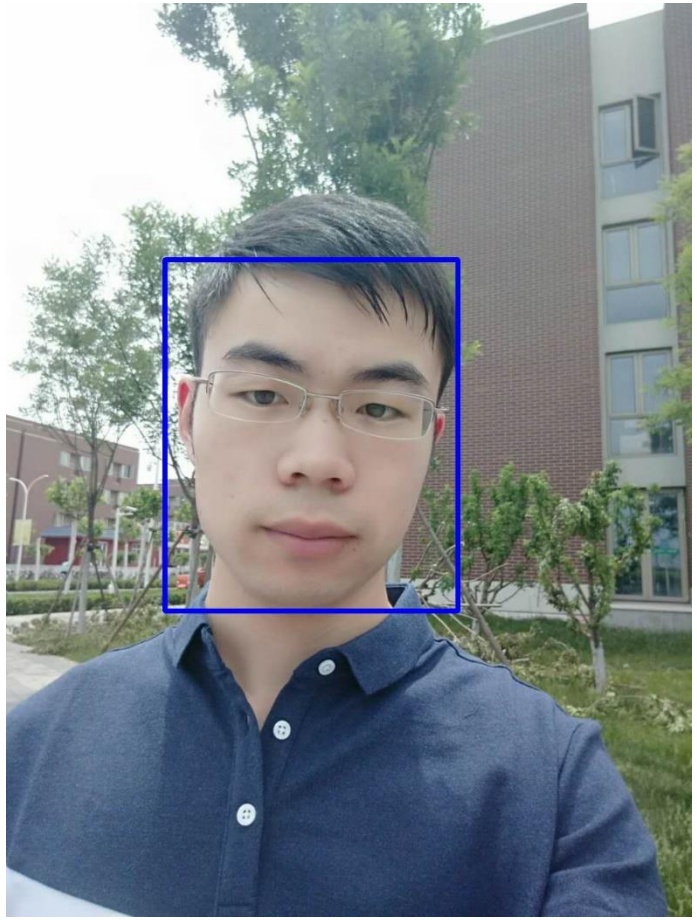
人脸检测：

```
// 构造人脸检测器
seeta::FaceDetector FD(seeta::ModelSetting("model/fd_2_00.dat"));
// 加载待识别图片
seeta::cv::ImageData image = cv::imread("1.png");
// 检测人脸
SeetaFaceInfoArray faces = FD.detect(image);
```

图像格式：HWC,BGR

```
struct SeetaImageData
{
    int width;
    int height;
    int channels;
    unsigned char *data;
};
```

FaceDetector



FaceLandmarker

检测结果:

```
struct SeetaPointF
{
    double x;
    double y;
};
```

包含头文件:

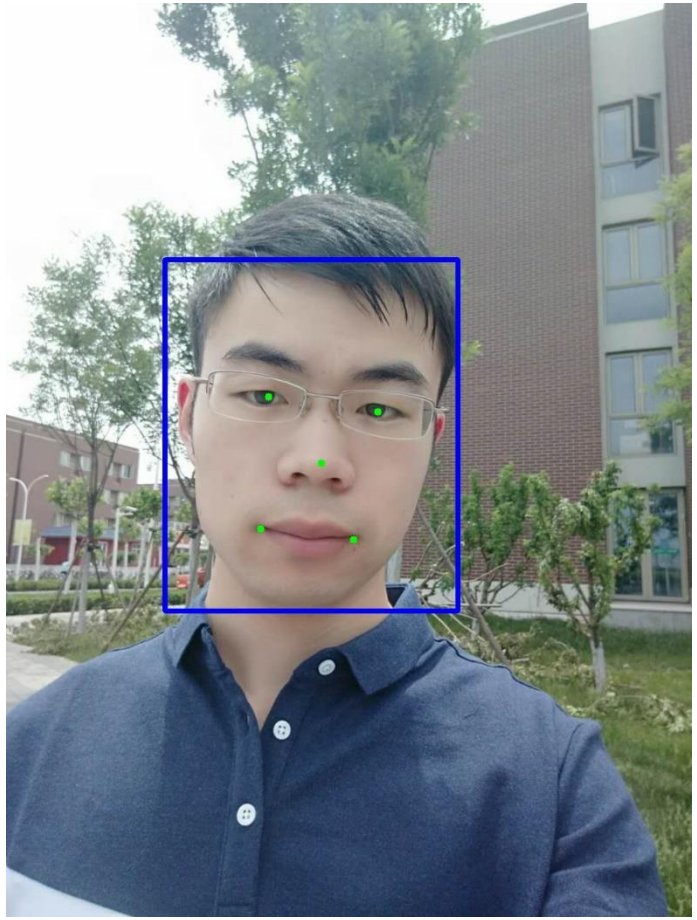
```
#include <seeta/FaceLandmarker.h>
#include <seeta/Struct_cv.h>
```

人脸关键点定位:

```
// 构造人脸关键点定位
seeta::FaceLandmarker FL(seeta::ModelSetting("model/pd_2_00_pts5.dat"));
// 加载待识别图片
seeta::cv::ImageData image = cv::imread("1.png");
// 检测人脸关键点
std::vector<SeetaPointF> points = FL.mark(image, faces.data[0].pos);
```

81点模型：pd_2_00_pts81.dat

FaceLandmarker



FaceDatabase

```
#include <seeta/FaceDatabase.h>
#include <seeta/Struct_cv.h>
```

```
// 构造识别库
seeta::FaceDatabase FDB(seeta::ModelSetting("model/fr_2_10.dat"));

// 1 vs 1 相似度对比
float similar_1vs1 = FDB.Compare(image1, points1, image2, points2);

// 1 vs N 注册底库
int id = FDB.Register(image, points);
// 注册失败返回 -1

// 1 vs N 识别
static const size_t topN = 2;
float similar[topN] = {0};
int64_t index[topN];
// 查询 top N 识别率
auto queryN = FDB.QueryTop(image, points, topN, index, similar);
// 返回识别数量，可小于 topN
```

FaceRecognizer

```
#include <seeta/FaceRecognizer.h>
#include <seeta/Struct_cv.h>
```

```
// 构造识别库
seeta::FaceRecognizer FR(seeta::ModelSetting("model/fr_2_10.dat"));

// 提取图片 1 特征
std::shared_ptr<float> features1(new float[FR.GetExtractFeatureSize()], std::default_delete<float[]>());
FR.Extract(image1, points1, features1.get());
// 提取图片 2 特征
std::shared_ptr<float> features2(new float[FR.GetExtractFeatureSize()], std::default_delete<float[]>());
FR.Extract(image2, points2, features2.get());
// 计算相似度
auto similar = FR.CalculateSimilarity(features1.get(), features2.get());
```

人脸识别示例

SeetaTech 中科视拓

- 注册底库照片

```
std::vector<std::string> GalleryImageFilename = { "1.jpg" };
std::vector<int64_t> GalleryIndex( GalleryImageFilename.size() );
for( size_t i = 0; i < GalleryImageFilename.size(); ++i )
{
    //register face into facedatabase
    std::string &filename = GalleryImageFilename[i];
    int64_t &index = GalleryIndex[i];
    std::cerr << "Registering... " << filename << std::endl;
    seeta::cv::ImageData image = cv::imread( filename );
    auto id = engine.Register( image );
    index = id;
    std::cerr << "Registered id = " << id << std::endl;
}
std::map<int64_t, std::string> GalleryIndexMap;
for( size_t i = 0; i < GalleryIndex.size(); ++i )
{
    // save index and name pair
    if( GalleryIndex[i] < 0 ) continue;
    GalleryIndexMap.insert( std::make_pair( GalleryIndex[i], GalleryImageFilename[i] ) );
}
```

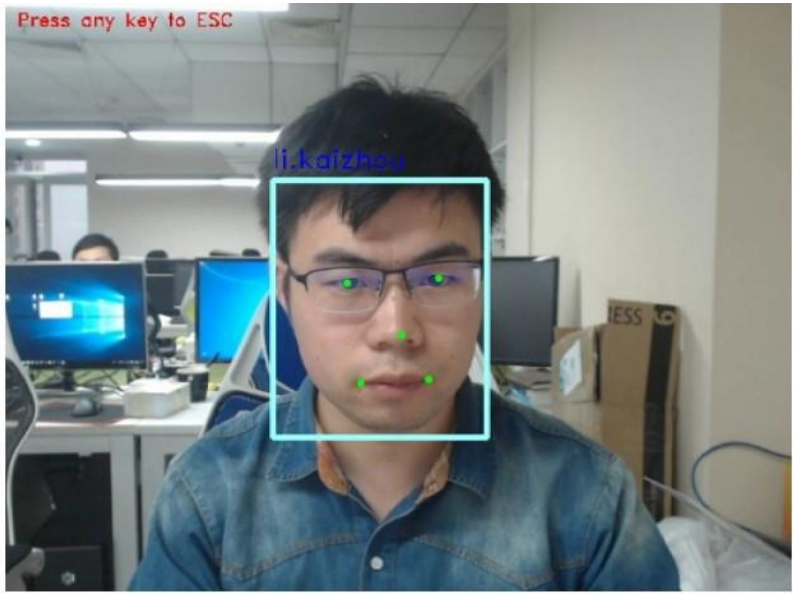
人脸识别示例

SeetaTech 中科视拓

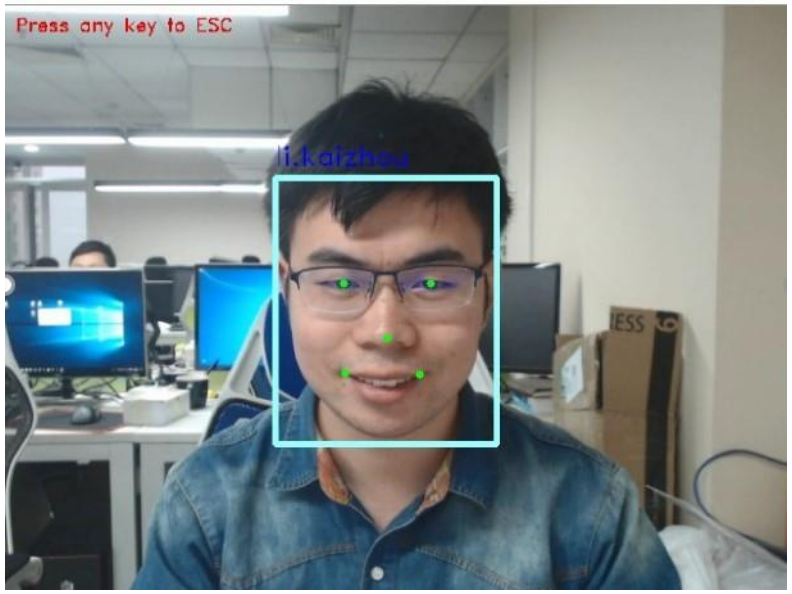
- 动态识别人脸

```
seeta::cv::ImageData image = frame;

// Detect all faces
std::vector<SeetaFaceInfo> faces = engine.DetectFaces( image );

for( SeetaFaceInfo &face : faces )
{
    // Query top 1
    int64_t index = -1;
    float similarity = 0;

    auto points = engine.DetectPoints(image, face);

    auto queried = engine.QueryTop( image, points.data(), 1, &index, &similarity );

    // no face queried from database
    if (queried < 1) continue;

    // similarity greater than threshold, means recognized
    if( similarity > threshold )
    {
        cv::putText( frame, GalleryIndexMap[index], cv::Point( face.pos.x, face.pos.y - 5 ), 3, 1, CV_RGB( 255, 128, 128 ) );
    }
}
```

人脸识别示例


Press any key to ESC
li.kaizhou

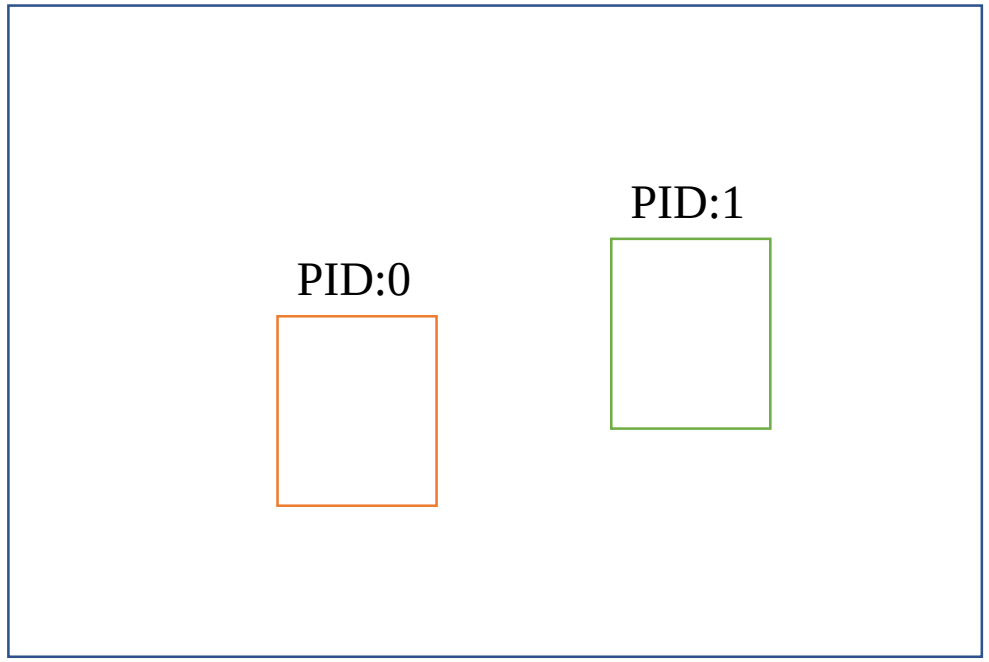

Press any key to ESC
li.kaizhou


Press any key to ESC
li.kaizhou

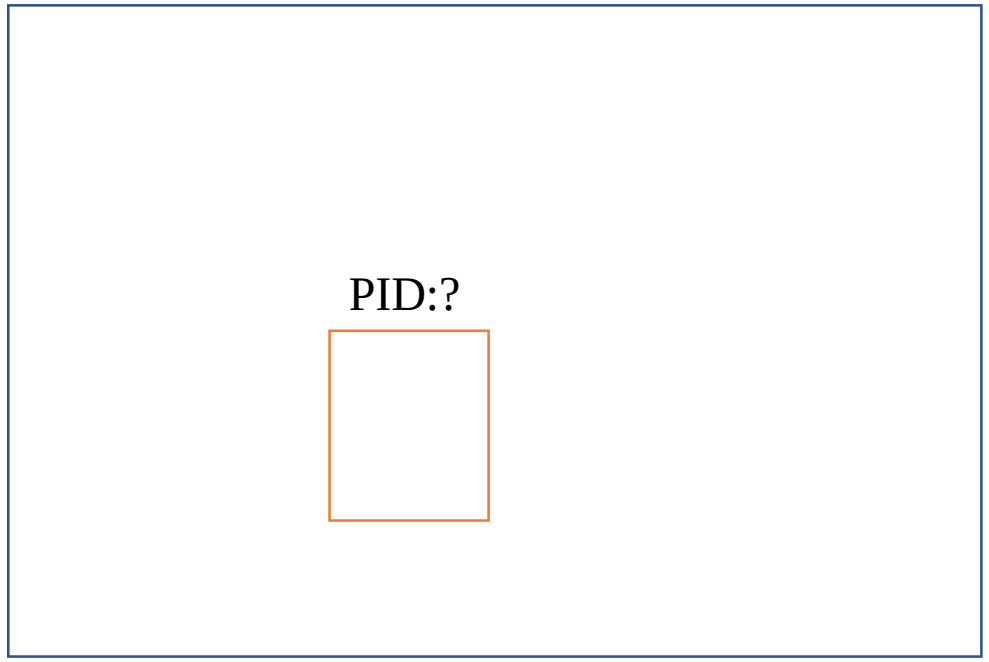
SeetaFace2应用举例

- 人脸跟踪、质量评估
- 人证对比
- 动态识别

人脸跟踪

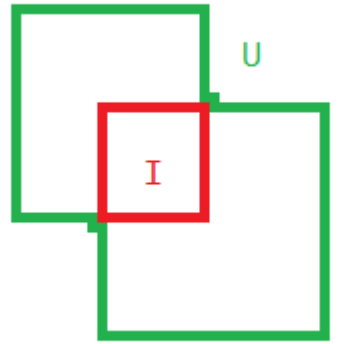


Frame: x



Frame: x+1

人脸跟踪




```
auto &face = faces[i];
for (auto &scored_tracked_face : scored_tracked_faces) {
    scored_tracked_face.iou_score = IoU(scored_tracked_face.face.pos, face);
}
if (scored_tracked_faces.size() > 1) {
    std::partial_sort(scored_tracked_faces.begin(), scored_tracked_faces.begin() + 1,
                      scored_tracked_faces.end(),
                      [](const ScoredTrackedFace &a, const ScoredTrackedFace &b) {
                          return a.iou_score > b.iou_score;
                      });
}
```

- 获取到当前第i张人脸和前一帧每张人脸IOU

人脸跟踪

SeetaTech 中科视拓

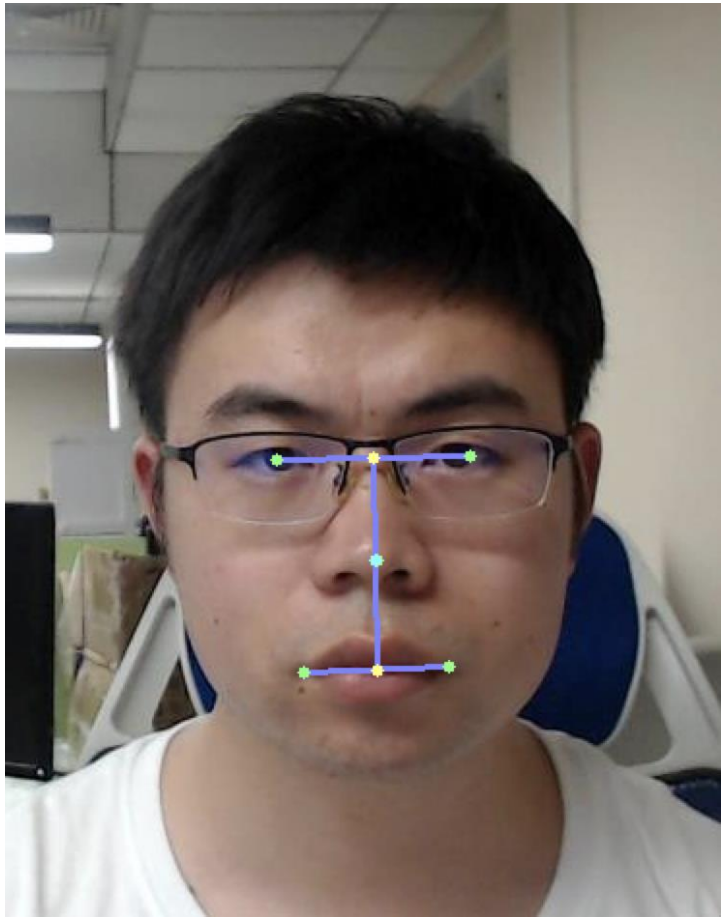
```
if (!scored_tracked_faces.empty() && scored_tracked_faces.front().iou_score > this->min_score) {
    ScoredTrackedFace matched_face = scored_tracked_faces.front();
    scored_tracked_faces.pop_front();
    TrackedFace &tracked_face = matched_face.face;
    if (matched_face.iou_score < max_score) {
        tracked_face.pos.x = (tracked_face.pos.x + face.x) / 2;
        tracked_face.pos.y = (tracked_face.pos.y + face.y) / 2;
        tracked_face.pos.width = (tracked_face.pos.width + face.width) / 2;
        tracked_face.pos.height = (tracked_face.pos.height + face.height) / 2;
    } else {
        tracked_face.pos = face;
    }
    tracked_face.conf = face_array.data[i].score;
    tracked_face.frame_no = frame_no;
    now_trakced_faces.push_back(tracked_face);
} else {
    TrackedFace tracked_face;
    tracked_face.pos = face;
    tracked_face.PID = max_PID;
    tracked_face.conf = face_array.data[i].score;
    tracked_face.frame_no = frame_no;
    max_PID++;
    now_trakced_faces.push_back(tracked_face);
}
```

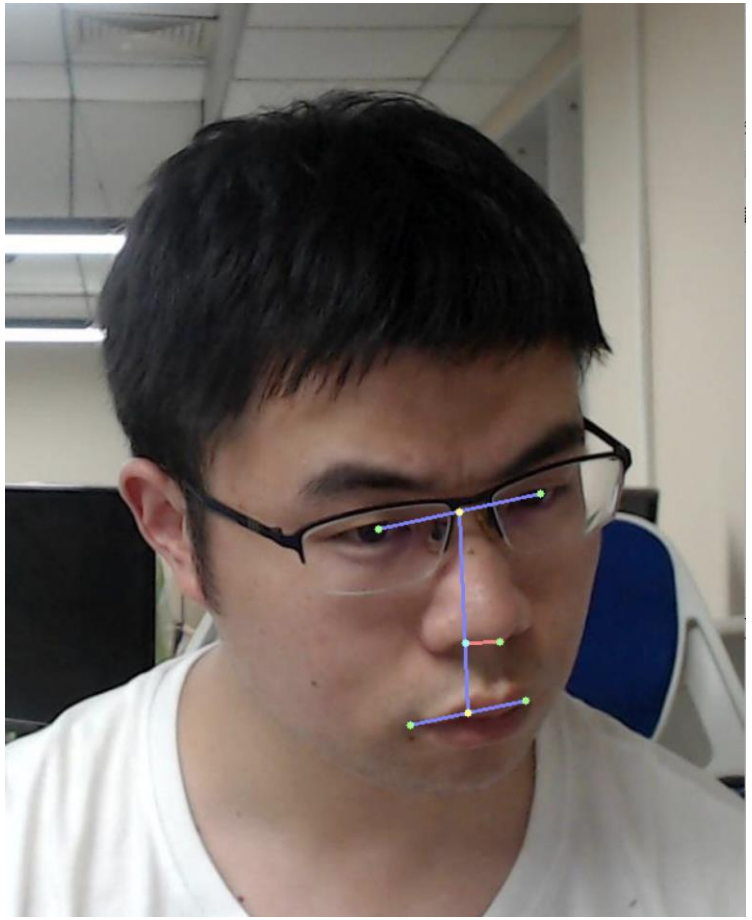
- IoU超过阈值则认为是同一张人脸

质量评估

- 人脸分辨率
- 人脸姿态
- 面部亮度
- 人脸清晰度

人脸姿态



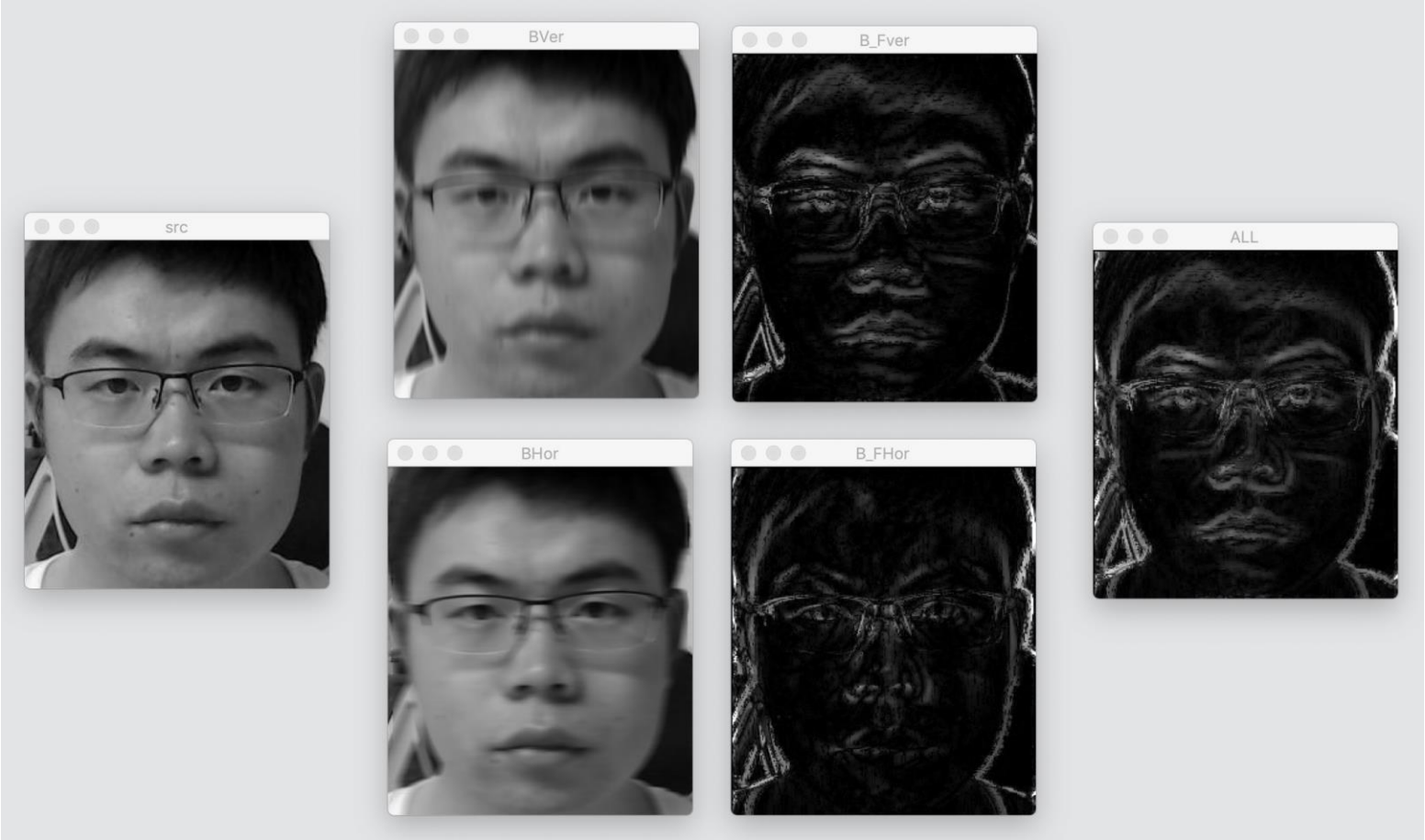


面部亮度

SeetaTech
中科视拓

```
cv::Mat image = cv::imread("1.png");
SeetaRect face = {10, 10, 100, 120};

cv::Mat patch = image(cv::Rect(face.x, face.y, face.width, face.height));
cv::Mat gray;
cv::cvtColor(image, gray, cv::COLOR_BGR2GRAY);
cv::Mat mean, std;
cv::meanStdDev(gray, mean, std);
auto brightness = mean.at<double>(0, 0);
```

人脸清晰度



- Reference: The Blur Effect: Perception and Estimation with a New No-Reference Perceptual Blur Metric

人脸清晰度

- Reference: The Blur Effect: Perception and Estimation with a New No-Reference Perceptual Blur Metric

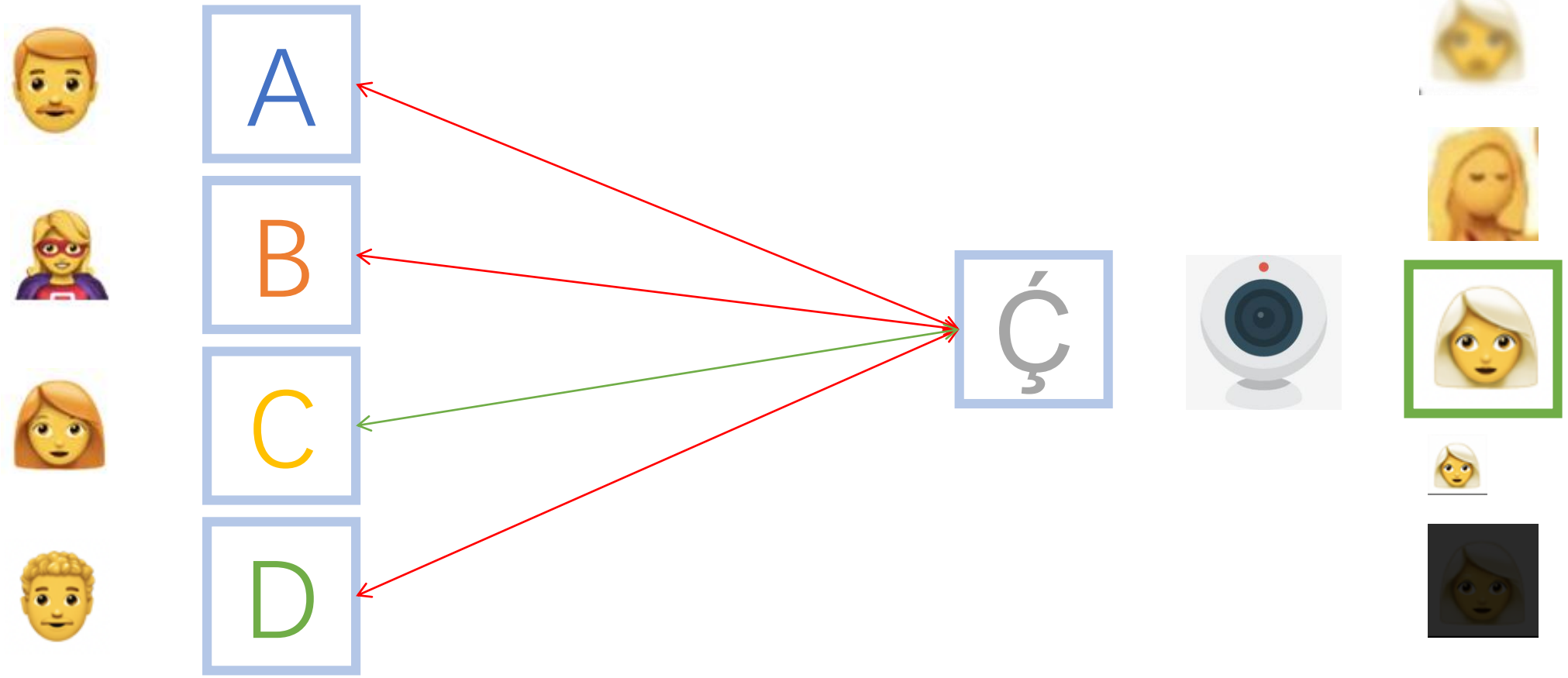
1比1识别

- 身份证小图（102x126）

1比N识别

SeetaTech
中 科 视 拓

PID:3



底库

抓拍

人脸识别应用

人脸认证

人脸闸机

动态识别

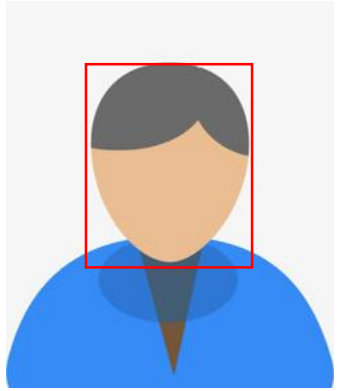
人脸检索

人脸门禁

人证对比

FAQ

- 身份证小图检测不到人脸
  - 标准化证件照
- 前端检测到人脸，服务器检测不到人脸
  - 特征差异
  - 图像处理
- 特征存储VS图片存储
  - 特征小而快速
  - 图片全而灵活

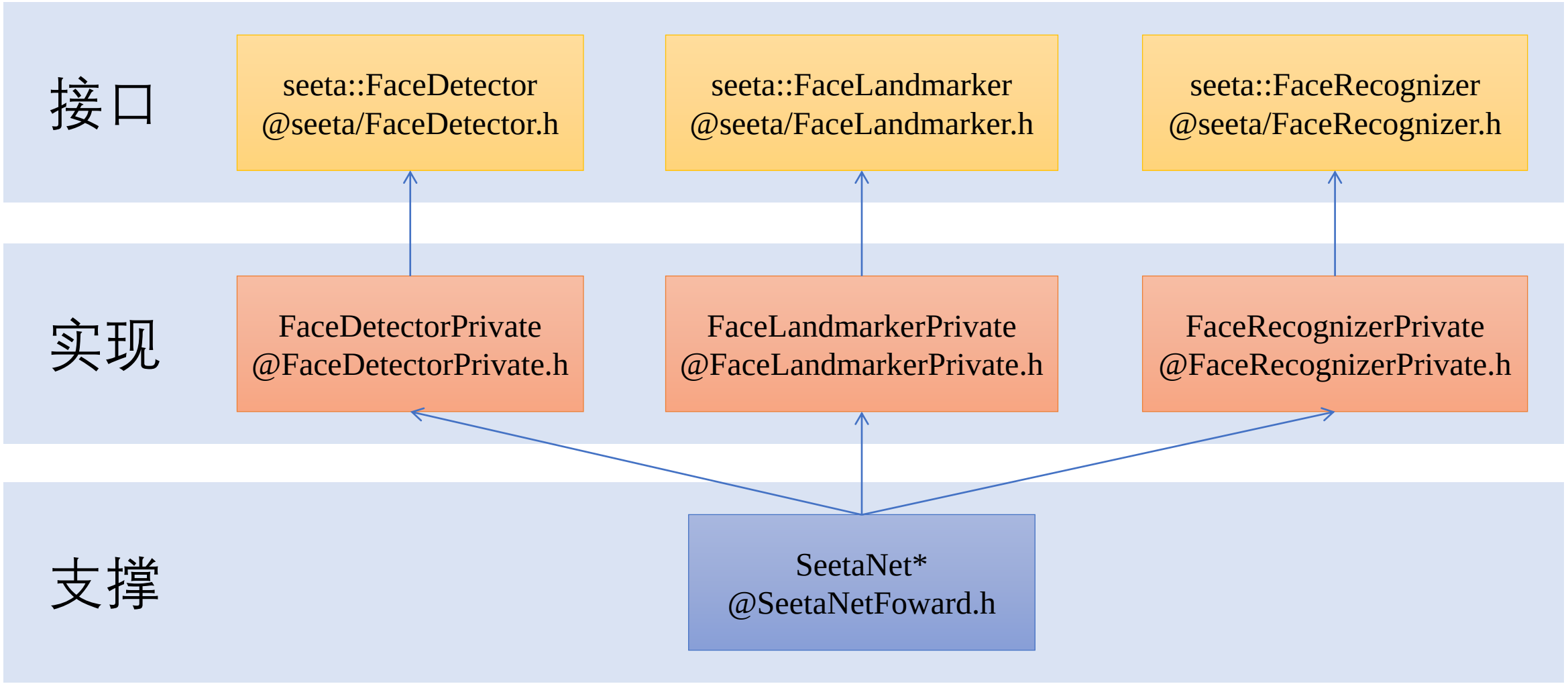


源码分析

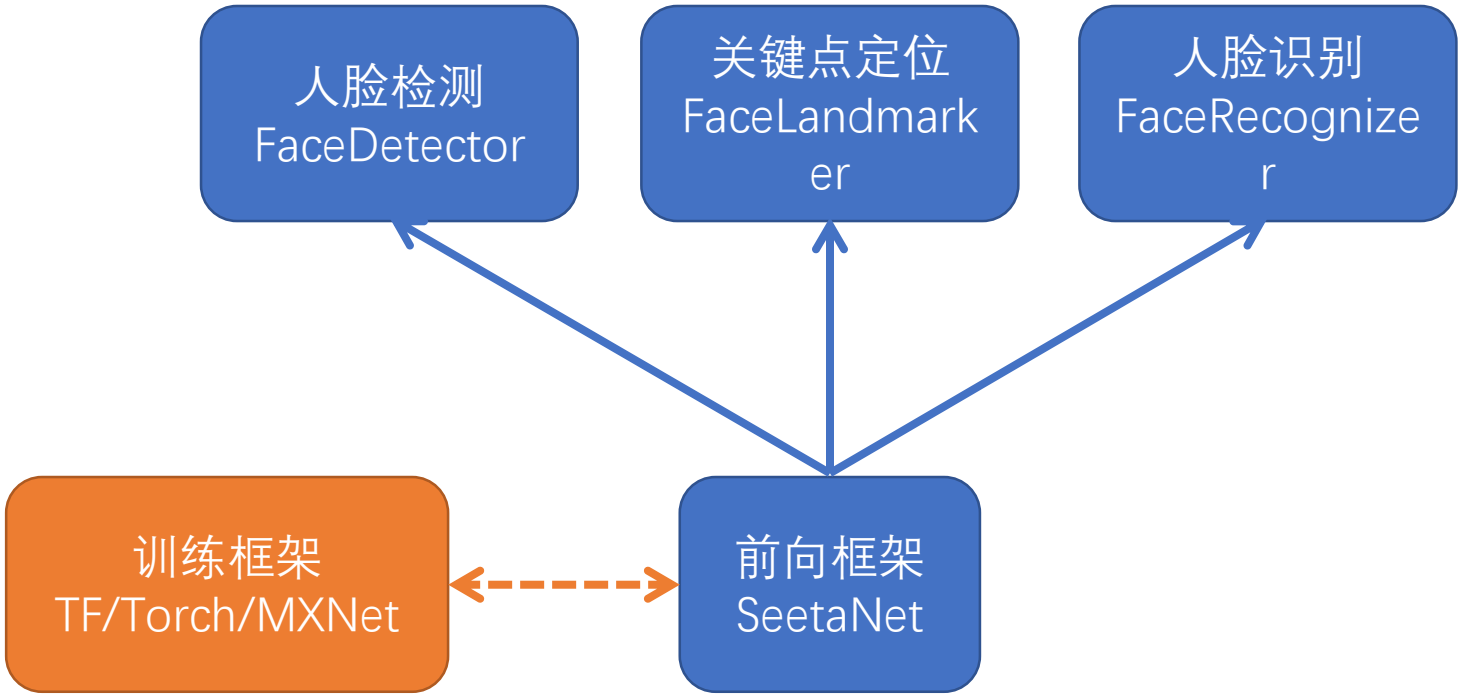
SeetaTech
中科视拓

- 1. 模型加载
- 2. 模型推理
- 3. 前处理
- 4. 后处理
- 5. 参数转换

源码结构


接口
seeta::FaceDetector
@seeta/FaceDetector.h
seeta::FaceLandmarker
@seeta/FaceLandmarker.h
seeta::FaceRecognizer
@seeta/FaceRecognizer.h
实现
FaceDetectorPrivate
@FaceDetectorPrivate.h
FaceLandmarkerPrivate
@FaceLandmarkerPrivate.h
FaceRecognizerPrivate
@FaceRecognizerPrivate.h
支撑
SeetaNet*
@SeetaNetFoward.h

模块划分


人脸检测
FaceDetector
关键点定位
FaceLandmarker
人脸识别
FaceRecognizer
训练框架
TF/Torch/MXNet
前向框架
SeetaNet

模型加载

- [SeetaNet/include/SeetaNetForward.cpp]

```
/**
* @brief Create the net from the given model configuration.
* @param [in] model The model generated by @c SeetaReadModelFromBuffer.
* @param [in] max_batch_size The max batch size you can feed.
* @param [in] process_device_type Chose the device running net. See @see SeetaNet_DEVICE_TYPE.
* @param [out] pnet A pointer pointing a @c SeetaNet_Net pointer. Returning an inner net structure.
* @return Return 0 only if everything is OK. May return error UNIDENTIFIED_LAYER
* @note Required to call @c SeetaReleaseNet with `net`, finalizing the inner net structure.
* @see SeetaReadModelFromBuffer SeetaNet_Net SeetaReleaseNet
*/
SEETANET_C_API int SeetaCreateNet(
        struct SeetaNet_Model *model, int max_batch_size, enum SeetaNet_DEVICE_TYPE process_device_type,
        struct SeetaNet_Net **pnet );
```

模型加载

- [SeetaNet/include/SeetaNetForward.cpp]

```
/**
 * @brief Create the net from the given model configuration.
 * @param [in] model The model generated by @c SeetaReadModelFromBuffer.
 * @param [in] max_batch_size The max batch size you can feed.
 * @param [in] process_device_type Chose the device running net. See @see SeetaNet_DEVICE_TYPE.
 * @param [out] pnet A pointer pointing a @c SeetaNet_Net pointer. Returning an inner net structure.
 * @param [out] pparam A pointer pointing a @c SeetaNet_SharedParam pointer. Returning an inner param structure.
 * @return Return 0 only if everything is OK. May return error UNIDENTIFIED_LAYER
 * @note Required to call @c SeetaReleaseNet with `net`, finalizing the inner net structure.
 * @see SeetaReadModelFromBuffer SeetaNet_Net SeetaReleaseNet
*/
SEETANET_C_API  int SeetaCreateNetSharedParam(
        struct SeetaNet_Model *model, int max_batch_size, enum SeetaNet_DEVICE_TYPE process_device_type,
        struct SeetaNet_Net **pnet, struct SeetaNet_SharedParam **pparam );
```

- [SeetaNet/src/SeetaNet.cpp] CreateNetSharedParam

```
pfun = CreateLayerMapCPU<NetF>::FindRunFunciton( layer_type );
SeetaNetBaseLayer<NetF> *tmp_layer = nullptr;
pfun( tmp_layer, *( ptmp_model->all_layer_params[i] ), output_net.tmp_NetResource );
tmp_layer->m_layer_type = layer_type;
```

- 网络初始化：逐层构造

模型加载

- [FaceDetector/seeta/FaceDetectorPrivate.cpp] Impl::LoadModelBuffer

```
SeetaReadModelFromBuffer( model_buffer12, size_t( buffer_lenght12 ), &model_[0] );
SeetaModelResetInput( model_[0], width_limit_, height_limit_ );
SeetaCreateNet( model_[0], 1, type, &net_[0] );

SeetaReadModelFromBuffer( model_buffer24, size_t( buffer_lenght24 ), &model_[1] );
SeetaCreateNet( model_[1], max_batch_size[0], type, &net_[1] );

SeetaReadModelFromBuffer( model_buffer48, size_t( buffer_lenght48 ), &model_[2] );
SeetaCreateNet( model_[2], max_batch_size[1], type, &net_[2] );
```

- 以人脸检测器为例：CascadeCNN 使用了三个子图

模型推理

- [SeetaNet/include/SeetaNetForward.h]

```
/**
 * @brief Feed the data @c SeetaNet_InputOutputData into the net, and do "Forward Propagation"
 * @param [in] net The net generated by @c SeetaCreateNet.
 * @param [in] counts Not used reserve parameter, 1 fro default.
 * @param [in] pinput_data The data feed the `net`
 * @return Return 0 only if everything is OK.
 * @note The `pinput_data->data_point_char` means the input data
 * @see SeetaNet_InputOutputData SeetaCreateNet
 */
SEETANET_C_API int SeetaRunNetChar( struct SeetaNet_Net *net, int counts, struct SeetaNet_InputOutputData *pinput_data );
```

- [SeetaNet/src/SeetaNet.cpp] RunNetTemplate

```
auto layer = output_net->Layer_vector[i];
return_result = layer->Process( bottom_blob_vector, top_blob_vector );
```

- 网络前项推理：逐层计算

模型推理

- [SeetaNet/include/SeetaNetStruct.h]

```
/**
* @brief The base data structure
*/
struct SeetaNet_InputOutputData
{
    float *data_point_float;    /**< Used in output mode, pointing to the specific blob */
    unsigned char *data_point_char;     /**< Used in input mode, pointing to image data */
    int number;                 /**< Number of the batch size */
    int channel;                /**< Number of the channels */
    int width;                  /**< Width of the blob (or input image) */
    int height;                 /**< Height of the blob (or input image) */
    int buffer_type;            /**< Not used reserve parameter, 0 for default (means local memory data)*/
};
typedef struct SeetaNet_InputOutputData SeetaNet_InputOutputData;
```

- SeetaNet输入输出结构

前处理

- [FaceRecognizer/seeta/FaceRecognizerPrivate.h]

```
bool FaceRecognizerPrivate::ExtractFeatureWithCrop(
        const SeetaImageData &srcImg,
        const SeetaPointF *llpoint,
        float *feats, uint8_t posNum )
{
    SeetaImageData dstImg;
    dstImg.width = GetCropWidth();
    dstImg.height = GetCropHeight();
    dstImg.channels = srcImg.channels;
    std::unique_ptr<uint8_t[]> dstImgData( new uint8_t[dstImg.width * dstImg.height * dstImg.channels] );
    dstImg.data = dstImgData.get();
    CropFace( srcImg, llpoint, dstImg, posNum );
    ExtractFeature( dstImg, feats );
    return true;
}
```

- 以人脸识别器为例：前处理包括了人脸对齐到256x256,后输入到网络提取特征

前处理

- [FaceLandmarker/seeta/FaceLandmarkerPrivate.h]
  FaceLandmarkerPrivate::PredictLandmark

```
if(
    !ResizeImage( src_img.data, src_img.width, src_img.height, src_img.channels,
                  dst_img.data, input_width_, input_height_, input_channels_ )
    ||
    !Predict( dst_img, landmarks, masks )
) return false;
for( auto &ite : landmarks )
{
    ite.x *= ( src_img.width - 1 );
    ite.y *= ( src_img.height - 1 );
}
```

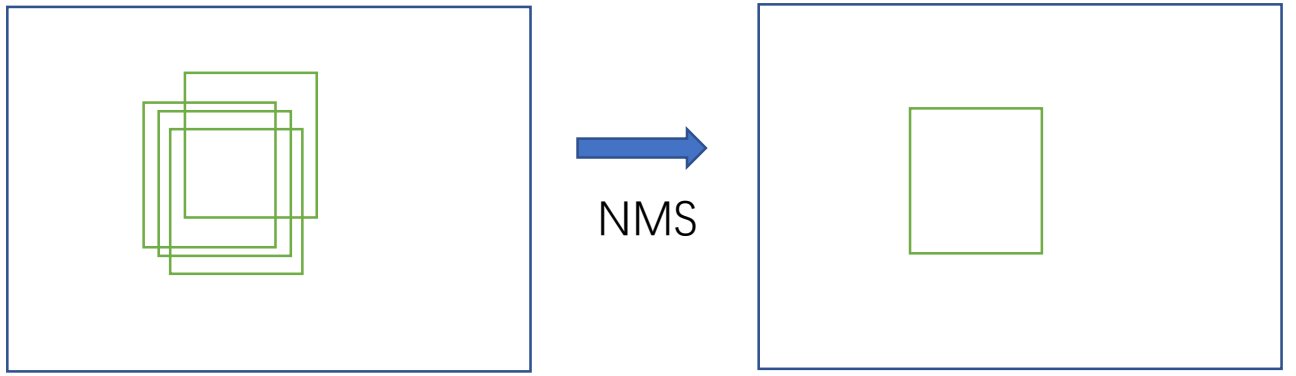
- 以关键点定位为例：前处理包括了图像的缩放，可选的颜色通道转换

后处理

- [FaceDetector/seeta/FaceDetectorPrivate.cpp] FaceDetectorPrivate::Detect

```
std::vector<Rect> winList;
winList = p->SlidingWindow( img, img_pad, p->net_[0], p->class_threshold_[0],
        local_min_face_size, local_max_face_size );
winList = p->NMS( winList, true, p->nms_threshold_[0] );

winList = p->RunNet( img_pad, p->net_[1], p->class_threshold_[1], 24, winList );
winList = p->NMS( winList, true, p->nms_threshold_[1] );

winList = p->RunNet( img_pad, p->net_[2], p->class_threshold_[2], 48, winList );
winList = p->NMS( winList, false, p->nms_threshold_[2] );
```

- 人脸检测器为例：每次检测过后，通过NMS得到最终的检测结果，然后进行逐级过滤。



参数转换

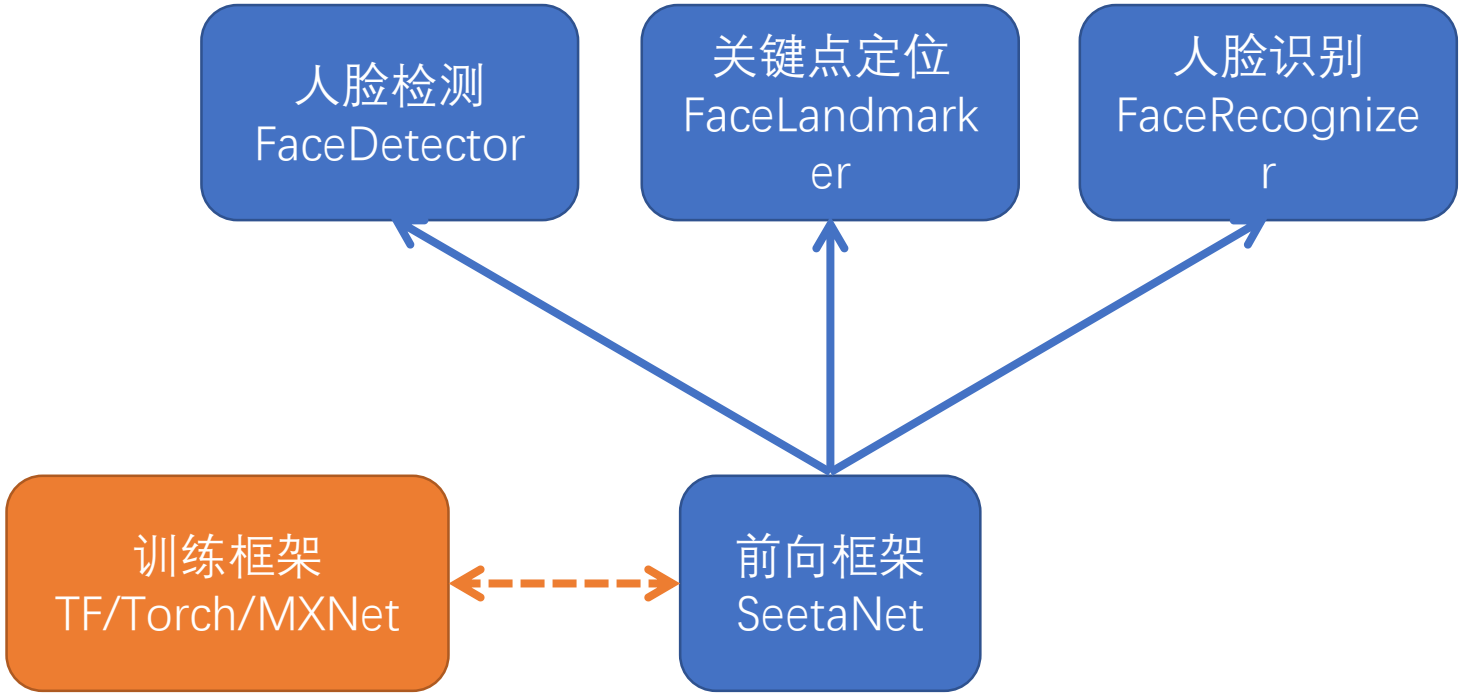
- [FaceLandmaker/seeta/FaceLandmarker.cpp] FaceLandmarkerPrivate::PointDetectLandmarks

```
CropFace( src_img.data, src_img.width, src_img.height, src_img.channels,
          dst_img.data, int( min_x ), int( min_y ), int( max_x ), int( max_y ) );

bool flag = PredictLandmark( dst_img, landmarks, masks );

for( int i = 0; i < landmark_num_; i++ )
{
    landmarks[i].x += min_x;
    landmarks[i].y += min_y;
}
```

- 人脸关键点为例：检测回归是基于裁剪出的图像的，最终要还原到原图坐标。

源码分析-总结

- 模型加载→参数转换→前处理→模型推理→后处理→参数转换→输出



源码分析-算法

SeetaTech
中科视拓

- 人脸检测算法：Cascade CNN，参考文献 A Convolutional Neural Network Cascade for Face Detection
- 特征点定位算法：FEC-CNN，参考文献 Robust FEC-CNN: A High Accuracy Facial Landmark Detection System
- 人脸特征提取：ResNet，参考文献 Deep Residual Learning for Image Recognition
- 人脸特征对比：向量余弦相似度计算

职业分享

- AI工程师：算法工程师+软件工程师

算法工程师

- 扎实基础
  - 机器学习、优化、图像处理、神经网络等等。
- 熟悉两种（以上）深度学习框架
  - 跳脱出框架限制
- 算法复现
  - 搭建轻框架
- 坚持阅读
  - 掌握最新的技术

软件工程师

SeetaTech
中科视拓

- 扎实基础
  - 计算理论、数据结构、算法、操作系统、软件工程、21天精通XXX等等
- 熟悉两种（以上）深度学习框架
  - 了解常用部署方式（tflite、torchscript、tvm等）
- 算法复现
  - 算法→部署，掌握算法特性
- 坚持阅读

QA

SeetaTech
中 科 视 拓

Thanks.