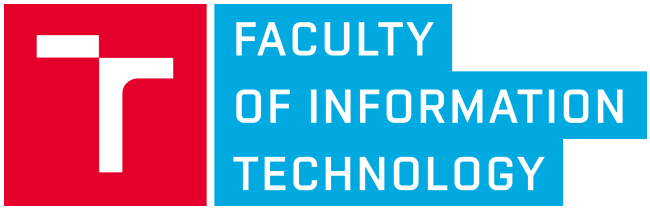

User Guide
SDK for object detection and anonymization

Výsledek Specializované algoritmy pro úpravu obrazu projektu TH04010394 Progresivní algoritmy pro zpracování obrazu

Michal Hradiš

Vysoké učení technické v Brně

Brno 2021

This document was created with financial support of TA ČR in project Progressive Image Processing Algorithms TH04010394.

Code and name of the project:

TH04010394	Progresivní algoritmy pro zpracování obrazu

Description

The SDK is provided as dynamically linked libraries for Windows. Both 32-bit and 64-bit versions are available. The interface is pure C. SDK can be used to detect and precisely localize objects in images and video which can be used for anonymization, statistics, retrieval, search and similar tasks. The SDK includes pretrained detectors for human faces and car license plates. Both detectors can handle a large range of poses, illumination conditions and image degradations (saturation, blur, compression). Additionally a model which is able to precisely localize 68 facial landmarks can be used as an automatic pre-processing step in portrait manipulation tools.

Requirements

Windows
SSE 2 instruction set.

Detection representation

```
struct FaceDetection {
        int id;
        float score;
        float left, top, right, bottom;
        float px0, px1, px2, px3, px4;
        float py0, py1, py2, py3, py4;
};
```

- *id* - unique object identifier
- *score* - detection confidence from 0 to 1.
- *left, top, right, bottom* - axis-aligned object bounding rectangle
- *px0, …, py4* - x,y coordinates of precise object location corners

Detector initialization and cleanup

```
extern "C" CNNDETECTOR_API void * create(const char *det1, const char *det2, const char *det3);
```

Description: Initializes detector for further use. Detectors are stored as separate binary files which contain neural networks. Single object detector can be composed of up to three neural networks which are run in a sequence. The returned pointer is managed by the library and has to be released using *release()*.

Configuration for face detection:

- *det1* - content of “bin_det1.xnormodel”
- *det2* - content of “bin_det2.xnormodel”
- *det3* - content of “bin_det3.xnormodel”

Configuration for license plate detection:

- *det1* - content of “lp_NET_005_c08_2x_44.bin”
- *det2* - NULL
- *det3* - NULL

Error behavior: Returns NULL on any error

```
extern "C" CNNDETECTOR_API void release( void * det);
```

Description: Releases all resources of a detector. The pointer is no longer valid after this call.

Detector usage

```
extern "C" CNNDETECTOR_API int detect( void * detector, unsigned char
*img, int width, int height, int rowStep, float detMinSize, float
thresholdScale, int threadCount, int *count, FaceDetection
**detections );
```

Description: Detects objects in an image using the specified detector.

Parameters:

- *detector* - Pointer representing a detector (returned by *create()*)
- *img* - Pointer to image data. Color channel order is BGR (24 bits per pixel)
  - Memory order is channels first, rows second.
  - The image has to be an array accessible by:
  - B = *img*[0 + x * 3 + y * *rowStep*]
  - G = *img*[1 + x * 3 + y * *rowStep*]
  - R = *img*[2 + x * 3 + y * *rowStep*]
- *width* - Number of columns of the image
- *height* - Number of rows of the image
- *rowStep* - Distance (in bytes) between image rows. Negative value flips the image vertically.
- *detMinSize* - Minimum size (height) of objects to be detected. Detection is faster for larger sizes. Even objects slightly smaller than *detMinSize* value get detected, but it is not guaranteed.
- *thresholdScale* - Inner detector sensitivity. Use values between 0.7 and 1.3 Higher values result in faster detection, but fewer detected faces. Lower values result in slower detection, but more detected faces. Default value is 1.

Returned results:

- *count* - Is filled with a number of detected faces.

- *detections* - Is filled with a pointer to detections. The pointer must be released using *releaseDetections()*.

Error behavior: Function returns error codes.
- 0 - ALL OK
- 1 - provided *detector* is NULL
- 2 - img is NULL
- 3 - detections is null

```
extern "C" CNNDETECTOR_API void releaseDetections( FaceDetection *detections );
```
Description: Releases detection pointer allocated by *detect()*. This function must be called after detect to avoid memory leaks.

Facial landmark localization

Facial landmarks are represented as *FaceKeypoints* which contain pairs of x,y coordinates of each of 70 facial landmarks.
```
    struct FaceKeypoints {
        float x[70];
        float y[70];
    };
```

```
extern "C" CNNDETECTOR_API void * createKeypointDetector( const char *det);
```
Description: Initializes the detector for further use. Needs a neural network represented in binary format as input. The returned pointer is managed by the library and has to be released using *releaseKeypointDetector()*.
Parameters:
- *det* - content of “bin_facial_keypoints_nchw.xnor”

Error behavior: Returns NULL on any error

```
extern "C" CNNDETECTOR_API void releaseKeypointDetector( void *det);
```
Description: Releases all resources of a facial landmark detector. The pointer is no longer valid after this call.

```
extern "C" CNNDETECTOR_API int detectKeypoints( void * detector, unsigned char *img, int width, int height, int rowStep, FaceDetection detection, FaceKeypoints *outKeypoints);
```
Description: Detects facial landmarks of a specific face represented by *detection* in an image using the specified keypoint detector.
Parameters:
- *img* - Pointer to image data. Color channel order is BGR (24 bits per pixel)

- Memory order is channels first, rows second.
- The image has to be an array accessible by:
- B = *img*[0 + x * 3 + y * *rowStep*]
- G = *img*[1 + x * 3 + y * *rowStep*]
- R = *img*[2 + x * 3 + y * *rowStep*]

- *width* - Number of columns of the image
- *height* - Number of rows of the image
- *rowStep* - Distance (in bytes) between image rows. Negative value flips the image vertically.
- *detection* - Detection returned by *detect().*

Returned results:

- *outKeypoints* - Pointer to a *FaceKeypoints* structure which will be filled with localized keypoints.

Error behavior: Function returns error codes.

- 0 - ALL OK
- 1 - provided *detector* is NULL
- 2 - *img* is NULL
- 3 - *outKeypoints* is null