MASKCNN_Masking_Library readme

Project has been compiled as a dynamically linked library. It has been developed and tested in a WSL virtual machine. It does not use gpu for inference, because at the time of development onnxruntime did not support gpu acceleration in WSL environment.

Dependencies:

Required dependencies are:

- onnxruntime
- opencv

Instructions for building onnxruntime for inference:
https://onnxruntime.ai/docs/build/inferencing.html

Instructions for building opencv:
https://github.com/Eemilp/install-opencv-on-wsl

Required lines in Cmakelists.txt:

```
find_package(OpenCV REQUIRED)
find_package(onnxruntime REQUIRED)
set(onnxruntime_INCLUDE_DIRS "/usr/local/include/onnxruntime")
set(onnxruntime_LIBRARIES "/usr/local/lib/libonnxruntime.so")
```

Example working Cmakelists.txt:

```
# Minimum CMake version
cmake_minimum_required(VERSION 3.10)

# Project name
project(TestLinking VERSION 1.0)

# Set C++ standard
set(CMAKE_CXX_STANDARD 20)
set(CMAKE_CXX_STANDARD_REQUIRED True)

find_package(OpenCV REQUIRED)
find_package(onnxruntime REQUIRED)
set(onnxruntime_INCLUDE_DIRS "/usr/local/include/onnxruntime")
set(onnxruntime_LIBRARIES "/usr/local/lib/libonnxruntime.so")

# Specify the include directory for the header files
include_directories(${CMAKE_SOURCE_DIR}/include ${CMAKE_SOURCE_DIR}/
${onnxruntime_INCLUDE_DIRS})

# Define the executable
add_executable(test_link main.cpp)

# Link the shared library to the executable
target_link_libraries(test_link PRIVATE ${CMAKE_SOURCE_DIR}/include/libmasking.so
${OpenCV_LIBS} ${onnxruntime_LIBRARIES})

# Optional: Set the output directory for the executable
set_target_properties(test_link PROPERTIES
    RUNTIME_OUTPUT_DIRECTORY ${CMAKE_SOURCE_DIR}/bin
)
```

Endpoints of the library:

```
class EXPORT Masking
{
public:
    static void mask_and_display(int label, std::string model_path,
std::string image_path, int sharpening_strength);
    static void mask_and_display(int label, std::string model_path, cv::Mat
image, int sharpening_strength);
    static cv::Mat final_mask(int label, std::string model_path, std::string
image_path, int sharpening_strength);
    static cv::Mat final_mask(int label, std::string model_path, cv::Mat
image, int sharpening_strength);
};
```

Method **mask_and_display** displays masked image using opencv.
Method **final_mask** returns the mask created by the program as cv::Mat object.

Methods parameters:

- **int label** - is the label of a class in the COCO dataset https://tech.amikelive.com/node-718/what-object-categories-labels-are-in-coco-dataset/
- **std::string model_path** is the filepath to the .onnx file containing the used model. Working model can be obtained by running downloadModel.py script
- **std::string image_path** is the filepath to an image that can be used for creating the mask and masking
- **cv::Mat image** is an image loaded using opencv in HWC (heigth width channel) format, of CV_8UC3 datatype CV_8UC3
- **int sharpening_strength** is an additional parameter used instead of thresholding for reducing the masked area. It should be in range 0 to 255

Classes:

Project contains three classes:

- **ImageOperations** - contains operations on opencv library
- **OrtOperations** - contains operations on onnxruntime
- **Masking** - is the library interface