Инструкция использования библиотеки

1) Открыть терминал

2) Перейти в удобный каталог: cd testing

3) Склонировать репозиторий: git clone

https://github.com/ChervyakovLM/FaceMetric.git

Для проверки **заданной библиотеки биометрической верификации** необходимо реализовать класс наследник от FACEAPITEST::Interface.

```
class Interface {
public:
    virtual ~Interface() {}

    virtual ReturnStatus
    initialize(const std::string &configDir) = 0;

    virtual ReturnStatus
    createTemplate(
        const Multiface &faces,
        TemplateRole role,
        std::vector<uint8_t> &templ,
        std::vector<EyePair> &eyeCoordinates,
        std::vector<double> &quality) = 0;

    virtual ReturnStatus
    matchTemplates(
        const std::vector<uint8_t> &verifTemplate,
        const std::vector<uint8_t> &initTemplate,
        double &similarity) = 0;

    virtual ReturnStatus
```

```
    train(
        const std::string &configDir,
        const std::string &trainedConfigDir) = 0;

    static std::shared_ptr<Interface>
    getImplementation();
};
```

Класс наследник должен содержать реализацию следующих функций:

initialize - инициализация алгоритма вычисления биометрических шаблонов;

createTemplate - вычисление шаблона;

matchTemplates - сравнение шаблонов;

train - донастройка алгоритма вычисления биометричесекийх шаблонов;

getImplementation – получение указателя на реализацию.

Пример реализации класса наследника приведён в файлах face_api_example_V.h и face_api_example_V.cpp, содержащиеся в каталогах include и src соответсвенно.

Для проверки **заданной библиотеки биометрической идентификации** необходимо реализовать класс наследник от FACEAPITEST::IdentInterface.

```
class IdentInterface {
public:
    virtual ~IdentInterface() {}

    virtual ReturnStatus
    initializeTemplateCreation(
        const std::string &configDir,
        TemplateRole role) = 0;

    virtual ReturnStatus
    createTemplate(
        const Multiface &faces,
        TemplateRole role,
```

```
    std::vector<uint8_t> &templ,
    std::vector<EyePair> &eyeCoordinates) = 0;

virtual ReturnStatus
finalizeInit(
       const std::string &configDir,
    const std::string &initDir,
    const std::string &edbName,
    const std::string &edbManifestName) = 0;

virtual ReturnStatus
initializeIdentification(
    const std::string &configDir,
    const std::string &initDir) = 0;

virtual ReturnStatus
identifyTemplate(
    const std::vector<uint8_t> &idTemplate,
    const uint32_t candidateListLength,
    std::vector<Candidate> &candidateList,
    bool &decision) = 0;

virtual ReturnStatus
galleryInsertID(
    const std::vector<uint8_t> &templ,
    const std::string &id) = 0;

virtual ReturnStatus
galleryDeleteID(
    const std::string &id) = 0;

static std::shared_ptr<IdentInterface>
```

```
    getImplementation();
};
```

Класс наследник должен содержать реализацию следующих функций:

initializeTemplateCreation - инициализация алгоритма вычисления биометрических шаблонов;

createTemplate - вычисление шаблона;

finalizeInit - создание индекса из всех шаблонов;

initializeIdentification - инициализация алгоритма поиска по индексу;

identifyTemplate – поиск по индексу;

galleryInsertID - добавление шаблона в индеск;

galleryDeleteID - удаление шаблона из индекса;

getImplementation – получение указателя на реализацию.

Пример реализации класса наследника приведён в файлах face_api_example_I.h и face_api_example_I.cpp, содержащиеся в каталогах include и src соответсвенно.