**项目说明：**

config/weights存放模型权重 config/config.yml 存放配置信息 config/warm\_up.jpg是用于预热模型的输入图片 config.config.get\_config 返回配置信息

model 包存放模型功能代码

init/initialize.py 用来初始化Milvus数据库中的表(collection)

milvus\_tool包存放milvus数据库相关操作

utils包存放使用到的工具类

service/face\_service 是人脸相关功能的接口

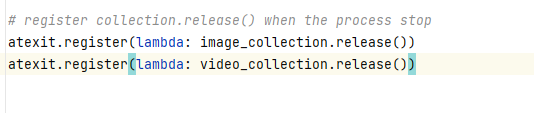
**使用说明：**

因为模型的加载，Milvus的连接和加载都需要在程序运行时保持状态，所以在程序时启动时需要加载模型和Milvus，在程序结束时要释放Milvus。以fastapi框架为例:

加载模型和Milvus：



释放Milvus：



在使用中模型model, 图片人脸库image\_collection 和 视频人脸库video\_collection作为参数传入各个接口函数。

**接口说明：**

1. 批量获取对齐的人脸

def get\_align\_faces\_batch(model: Face\_Onnx, paths,  
 enhance=False, confidence=0.99):

|  |  |  |
| --- | --- | --- |
| 传入参数 | 参数 | 参数说明 |
| model | 模型 |
| paths | 图片路径数组，类型为list |
| enhance | 是否对提取出的人脸增强 |
| confidence | 提取的人脸置信度阈值，设置在0.9以上，设置的值高可以过滤低质量的人脸 |
| 返回 | 无 | 返回类型为list的list，例如：  [ [ 人脸1，人脸2，... ], [ ... ] ... [ ... ] ]  list中的每个list存放一个图片中出现的对齐后的人脸 (可能不止一个)，若为出现人脸则该list为空[ ]。每个人脸为numpy数组，是opencv的图片表示格式( 维度为(h, w, 3), 颜色格式为bgr) |

1. 批量获取图片中的人脸向量

def get\_face\_embeddings(model: Face\_Onnx, paths, aligned=False,  
 enhance=False, confidence=0.99, merge=False):

|  |  |  |
| --- | --- | --- |
| 传入参数 | 参数 | 参数说明 |
| model | 模型 |
| paths | 图片路径数组，类型为list |
| enhance | 是否对提取出的人脸增强 |
| confidence | 提取的人脸置信度阈值，设置在0.9以上，设置的值高可以过滤低质量的人脸 |
| merge | 是否将所有图片的人脸向量放在同个数组里，默认为False，在提取一个视频关键帧时设置为True， |
| 返回 | 无 | 返回类型为list的list，例如：  [ [ 向量1，向量2，... ], [ ... ] ... [ ... ] ]  list中的每个list存放一个图片中出现的人脸向量(可能不止一个)，若为出现人脸则该list为空[ ]  在merge为True时，返回的为list，在提取视频关键帧的人脸向量时使用，表示同个视频的所有人脸向量。 |

1. 人脸向量批量插入Milvus

def add\_embeddings2milvus(collection, faces, flush=False):

备注：限制了人脸id的长度为50，当有一个人脸id的长度大于50时，不会执行插入返回为空列表[ ]

|  |  |  |
| --- | --- | --- |
| 传入参数 | 参数 | 参数说明 |
| collection | 操作的milvus中的collection(表)，图片人脸表或者视频人脸表 |
| faces | 人脸数组，类型为list，list中的每个单元为dict:{“id”: ... ,”embedding”: ...}，id是标识这个人脸的id（图片id或者视频id），类型为str，长度小于50；embedding是这个人脸的特征向量 |
| flush | 在插入完之后是否刷新，数据插入后，可以马上被检索到，但是不会马上写入磁盘，当写入的向量可以形成一个簇或者经过一定的时间后才会写盘，建议在大规模插入向量时flush设置为True |
| 返回 | 无 | 返回类型为list，list中的每个单元为dict：  {"primary\_key": 人脸向量的id , 'isSuccess':True} )  由于是批量插入人脸向量，每次执行插入语句可以最大插入5000条向量，所以暂时无法判断是否插入失败(没遇到过插入失败的情形) |

1. 按向量id批量删除向量

def delete\_face\_by\_primary\_key\_batch(collection, primary\_keys)

|  |  |  |
| --- | --- | --- |
| 传入参数 | 参数 | 参数说明 |
| collection | 操作的milvus中的collection(表)，图片人脸表或者视频人脸表 |
| primary\_keys | 向量id数组类型为[ ] |
| 返回 | 无 | 返回删除失败个数，类型为int  由于是批量删除人脸向量，只执行一次删除语句，所以暂时无法判断每一个向量是否删除失败 |

1. 按向量对应的人脸图片/视频id批删除向量

def delete\_face\_by\_object\_id(collection, object\_ids)

|  |  |  |
| --- | --- | --- |
| 传入参数 | 参数 | 参数说明 |
| collection | 操作的milvus中的collection(表)，图片人脸表或者视频人脸表 |
| object\_ids | 人脸图片或者视频id数组类型为[ ] |
| 返回 | 无 | 返回类型为list，list中的每个单元为dict：  {"id": 人脸图片或者视频id , 'isSuccess':True/False} ) |

1. 批量人脸搜索图片

def search\_face\_image(model: Face\_Onnx, collection, imgs,  
 enhance=False, score=0.5, limit=10, nprobe=50)

:

|  |  |  |
| --- | --- | --- |
| 传入参数 | 参数 | 参数说明 |
| model | 模型 |
| collection | 操作的milvus中的collection(表)，图片人脸表或者视频人脸表 |
| imgs | 图片数组，类型为list |
| enhance | 是否对人脸增强后在检索 |
| score | 人脸最低余弦相似度，取值[-1, 1], 建议设置在0.4以上，不要高于0.5 |
| limit | 返回满足相似度条件下最多数量的向量个数 |
| nprobe | 查询时检索的向量簇，默认为50， 值越大搜索的范围越大，时间越长 |
| 返回 | 无 | 返回类型为list的list，例如：  [ [ 结果1，结果2，... ], [ ... ] ... [ ... ] ]  list中的每个list存放一个人脸中匹配到的图片信息(可能不止一个)，若未匹配到则该list为空[ ]  每个结果的类型为dict:  {'id': 图片id,  'score': 相似度  } |

1. 批量人脸搜索视频

def search\_face\_video(model: Face\_Onnx, collection, imgs,  
 enhance=False, score=0.5, limit=10, nprobe=50)

:

|  |  |  |
| --- | --- | --- |
| 传入参数 | 参数 | 参数说明 |
| model | 模型 |
| collection | 操作的milvus中的collection(表)，图片人脸表或者视频人脸表 |
| imgs | 图片数组，类型为list |
| enhance | 是否对人脸增强后在检索 |
| score | 人脸最低余弦相似度，取值[-1, 1], 建议设置在0.4以上，不要高于0.5 |
| limit | 返回满足相似度条件下最多数量的向量个数，向量个数不等于视频个数，即便在视频人脸插入时经过人脸去重，也难免会有一个视频有多个相同人脸的情况 |
| nprobe | 查询时检索的向量簇，默认为50， 值越大搜索的范围越大，时间越长 |
| 返回 | 无 | 返回类型为list的list，例如：  [ [ 结果1，结果2，... ], [ ... ] ... [ ... ] ]  list中的每个list存放一个人脸中匹配到的视频信息(可能不止一个)，若未匹配到则该list为空[ ]  每个结果的类型为dict:  {'id': 视频id,  'score': 视频最匹配的人脸相似度  } |

1. 批量获取人脸图片的质量分数

def get\_face\_quality\_batch(model: Face\_Onnx, paths,  
 aligned=False):

|  |  |  |
| --- | --- | --- |
| 传入参数 | 参数 | 参数说明 |
| model | 模型 |
| paths | 图片路径数组，每张图片只能有一张人脸 |
| aligned | 是否为校准后的人脸 |
| 返回 | 无 | 返回分数数组，类型为list， 分数能在60以上就是质量很高的图片，检索用的人脸图片一般在40以上 |

1. 批量获取增强后的人脸图片

def enhance\_face\_batch(model: Face\_Onnx, paths,  
 aligned=False):

|  |  |  |
| --- | --- | --- |
| 传入参数 | 参数 | 参数说明 |
| model | 模型 |
| paths | 图片路径数组，每张图片只能有一张人脸 |
| aligned | 是否为校准后的人脸 |
| 返回 | 无 | 返回增强后的对齐的人脸图片数组，类型为list， 人脸图片为opencv的表示方式(维度(h, w, 3), 颜色格式为bgr) |

1. 批量提取视频关键帧

def extract\_key\_frames\_batch(paths)

|  |  |  |
| --- | --- | --- |
| 传入参数 | 参数 | 参数说明 |
| paths | 视频路径数组 |
| 返回 | 无 | 返回每个视频的关键帧信息，类型为list的list：  [ [帧信息1，帧信息2，... ], [ ... ] ... [ ... ] ]  每个帧信息格式为dict：  {  'frame': opencv格式下的帧图片表示,  'timestamp': 在视频中的视频，单位：秒 } |
|  |  |  |

1. 批量提取多个视频的所有关键帧中的人脸向量，包含同个视频中相同人脸的去重

def get\_videos\_face\_embedding(model: Face\_Onnx, video\_paths, enhance=False, confidence =0.99, threshold=0.5):

:

|  |  |  |
| --- | --- | --- |
| 传入参数 | 参数 | 参数说明 |
| model | 模型 |
| video\_paths | 不同视频的关键帧路径数组 |
| enhance | 是否对人脸增强后在检索 |
| confidence | 提取的人脸置信度阈值，设置在0.9以上，设置的值高可以过滤低质量的人脸 |
| threshold | 人脸去重的相似度阈值，默认0.5，建议0.4-0.5之间 |
| 返回 | 无 | 返回类型为list的list，例如：  [ [ 向量1，向量2，... ], [ ... ] ... [ ... ] ]  list中的每个list存放一个视频中出现的人脸向量，若为出现人脸则该list为空[ ] |

**Milvus安装：**

采用2.2.9版本

**单机版：**

因为之前的开发都是在自己主机上完成的，没有集群环境，所以用的Milvus是单机版。单机版使用docker运行，对应的文件在压缩包里面。

开启docker后，把压缩包中的docker-compose.yml 和milvus.yaml文件放到一个文件夹下，在该目录打开控制台 使用命令docker-compose up -d开启Milvus，使用命令docker-compose down关闭Milvus

https://milvus.io/docs/install\_standalone-docker.md

**集群部署：**

<https://milvus.io/docs/install_cluster-milvusoperator.md>

**项目环境：**

在压缩包的face.yml文件中