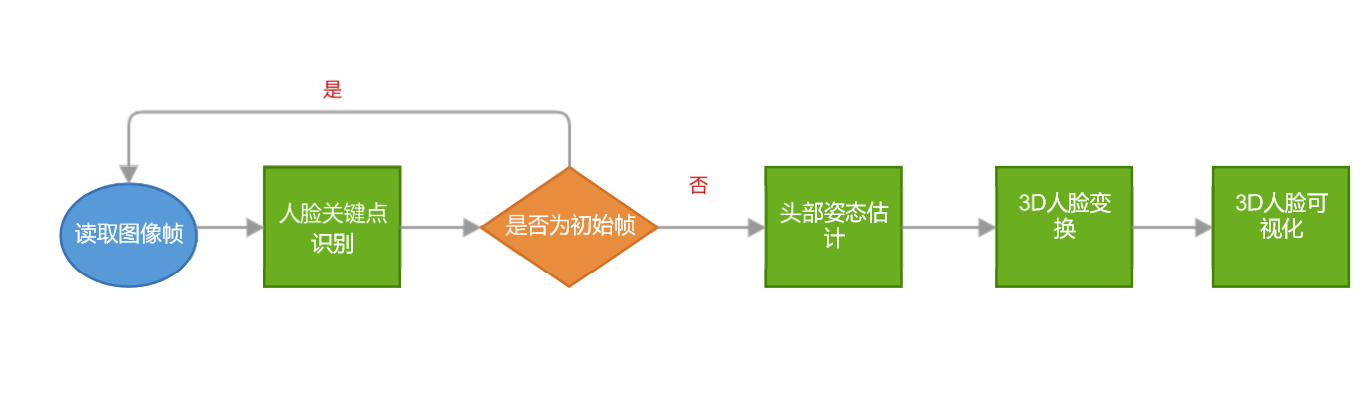
驾驶员头部姿态说明文档

一 驾驶员头部姿态估计算法说明

驾驶员头部姿态估计主要分为驾驶员头部关键点提取，驾驶员头部姿态估计，驾驶员头部3d显示。如下图所示：



1. 驾驶员头部关键点提取

主要利用dlib库的检测器和训练模型提取人脸68个关键点

1. 驾驶员头部姿态估计

主要利用2d-2d的对极几何的估计相机相邻两帧的变换情况

1. 驾驶员人脸3d显示

使用PCL库中3d点云三维显示。

二 依赖库安装

主要需要下载pcl,opencv4.,dlib.

1.安装dlib库：

cd dlib

mkdir build

cmake .. -DUSE\_AVX\_INSTRUCTION=ON -DUSE\_SSE4\_INSTRUCTIONS=ON

cmake --build . --config Release --target install

sudo make install

2.安装pcl库：

sudo add-apt-repository ppa:v-launchpad-jochen-sprickerhof-de/pcl  
sudo apt-get update  
sudo apt-get install libpcl-all

3.安装opencv

(1).安装依赖库

sudo apt-get install build-essential libgtk2.0-dev libvtk5-dev libjpeg-dev libtiff5-dev libjasper-dev libopenexr-dev libtbb-dev

(2).下载源码

https://opencv.org/releases/

(3).编译源码

cmake ..

make -j4

sudo make install

然后再使用release模式运行程序，程序就能加速运行

三 驾驶员头部姿态估计代码说明

1. 读入3d人脸数据与处理

* 读取标准人脸3d特征点：

Void ReadGeneralFaceShapePoints(const std::string filename, std::vector<double>& general\_face\_shape\_points);

传入的是存放人脸特征点的文件，该文件以（x1,y1,z1,x2,y2,z2,…,xn,yn,zn）T的格式保存。参数general\_face\_shape\_points是用来以vector格式存放3d特征点的变量作为输出。

* 读取标准人脸68个特征点的索引：

Void ReadGeneralFaceShapePointsIndex(const std::string filename,

std::vector<int>& face\_keypoints\_index);

传入两个参数 ：一个是存放人脸68个关键点索引的文件，以（1，2，3，…）T列向量保存。另一个是以vector格式存放68个关键点索引的变量作为输出。

* 数据归一化：

NormalizeFacePoints(const std::vector<double>& general\_face\_shape\_points, cv::Mat & general\_face\_shape\_points\_matrix);

数据归一化是因为3d人脸的数据的量纲较大，一开始使用Pnp算法估计头部姿态时，为了估计较准确减少跳动需要对数据进行归一化。目前对极几何算法可以不用数据归一化。

1. 分解本质矩阵，求R，t

Void DecomposeEssentialMatrix(const cv::Mat essential\_matrix, cv::Mat\_<double> &R1, cv::Mat\_<double> &R2, cv::Mat\_<double> &t1, cv::Mat\_<double> &t2);

传入的参数：essential\_matrix是求出的本质矩阵，R1，R2，t1，t2是分解出的四个解。

1. 利用三角化筛选R，t

double TestTriangulation(const std::vector<cv::Point2d>& lastframe, const std::vector<cv::Point2d>& curframe, cv::Mat\_<double> R, cv::Mat\_<double> t);

参数：lastframe为上一帧检测的人脸关键点，curframe为当前帧检测的人脸关键点，R，t为求解出的最终姿态。

返回：正深度占比

1. 人脸关键点检测

void FaceDetection(cv::Mat& inputimage, cv::Mat& outputimage,

dlib::frontal\_face\_detector& detector, dlib::shape\_predictor& pose\_model,

std ::vector<cv::Point2d>& face\_detect\_keypoints);

参数：inputimage为输入的图片，outputimage为输出的图片，detector为dlib库中的检测器，pose\_model为输入的人脸训练模型，face\_detect\_keypoints为输出的人脸检测的关键点。

1. 求解本质矩阵

void SolveRT(const std::vector<cv::Point2d>& face\_keypoint\_init, const

std::vector<cv::Point2d>& face\_keypoint\_cur, cv::Mat& transform\_matrix);

参数：face\_keypoint\_init为前一帧识别的关键点，face\_keypoint\_cur为后一帧识别的关键点，transform\_matrix为返回的变换矩阵。

1. 变换3d人脸

void TransformGeneralFacePoints(cv::Mat& transform\_matrix,

cv::Mat& general\_face\_shape\_points\_matrix);

参数：transform\_matrix为传入的变换矩阵，general\_face\_shape\_points\_matrix为返回的经变化后的3d人脸特征点。

1. 3d人脸显示

点云类pointcloud，只需在代码中定义一个点云就可以，在构造函数中会自动为其生成用于显示的点云。

成员函数：

void AddPointsToCloud(const cv::Mat& face\_3d\_final)：通过传入最终变换后的3d人脸特征点，加入到点云变量中。

PointCloud::Ptr GetCloud()：返回点云变量，主要用于显示的初始化

void ClearPointCloud()：清除点云中的点。

显示类viewer，构造函数需要其传入点云变量，以初始化显示类的成员变量。

void AddPointCloud()：添加点云

void UpdatePointCloud(int second)：更新点云，传入停留点云的秒数。