# 优化ceres/ceres.hpp：

**package：Ceres**

**lib：${CERES\_LIBRARIES}**

**namespace：ceres**

**struct CostFunctor{**

**...**

**template<typename T>**

**bool operator() (**

**const T \*x, T \*error**

**) const {...}**

**}**

|  |  |
| --- | --- |
| AutoDiffCostFunction<type\_functor, n\_out, n\_in...>(functor) | **自动微分代价函数** |
| Solve(options, \*problem, \*summary) | **启动**优化 |

|  |  |  |
| --- | --- | --- |
| Problem | | **最小二乘问题** |
| 方法 | AddResidualBlock(\*CostFunction, \*LossFunction, \*x) | **添加残差块** |

|  |  |  |
| --- | --- | --- |
| Solver::Options | | **求解**配置 |
| **属性** | **bool** minimizer\_progress\_to\_stdout | **最小化标准**输出进度 |
| **LinearSolverType** linear\_solver\_type | 增量方程求解**方法** |

|  |  |  |
| --- | --- | --- |
| Solver::Summary | | **求解**汇总 |
| 方法 | BriefReport / FullReport() | 信息 |

# 词袋DBoW3/DBoW3.h：

**lib：/usr/local/lib/libDBoW3.so**

**namespace：DBoW3**

**todo：https://github.com/rmsalinas/fbow**

**回环检测的阈值 (连续判定)：3 \* score(v[cur], v[cur - 1])**

|  |  |  |
| --- | --- | --- |
| Vocabulary | | 词汇表 **(k叉树)** |
| **形参** | k=10, L=5, weighting=TF\_IDF, scoring=L1\_NORM | 配置信息 |
| filename | 文件 |
| 方法 | create(**std::vector<cv::Mat>** training\_features) | 训练 |
| empty() | 空判断 |
| save("\*.yaml.gz") | 保存 |
| transform(**cv::Mat** features, **BowVector** out) | **特征 →** 词向量 |
| score(**BowVector** v1, **BowVector** v2) | 相似度 |

|  |  |  |
| --- | --- | --- |
| Database(**Vocabulary** voc, false) | | 数据库 |
| **方法** | add(**cv::Mat** features) | 添加**数据** |
| query(**cv::Mat** features, **QueryResults** out, max\_results) | 查询**相似数据** |

# 矩阵Eigen：

**package：Ceres**

**lib：${CERES\_LIBRARIES}**

**namespace：Eigen**

## 分解Cholesky：

**todo: .ldlt**

|  |  |  |
| --- | --- | --- |
| LDLT | | Cholesky分解器 **()** |
| **方法** | **transpositionsP()** | **排列矩阵** |
| **matrixL()** | **下三角矩阵** |
| **vectorD()** | **对角线向量** |
| **solve(b)** | **求解** |

## 内核Core：

|  |  |
| --- | --- |
| 宏： | |
| EIGEN\_MAKE\_ALIGNED\_OPERATOR\_NEW | **重写内存对齐**的**new**、delete运算 |

|  |  |
| --- | --- |
| l1CacheSize / l2CacheSize / l3CacheSize() | 返回**CPU缓存**大小 |
| aligned\_allocator<Type> | **内存分配**器 |

|  |  |  |
| --- | --- | --- |
| Vector<dtype, nr> / VectorXd | | **向量** |
| Matrix<dtype, nr=-1, nc=-1> / MatrixXd | | **矩阵** |
| Map<EigenT> | | **映射 (浅拷贝)** |
| 形参 | \*dtype | **深拷贝数组 / 矩阵** |
| 属性 | size() | 元素数 |
| rows / cols() | 行列数 |
| cast<dtype>() | 类型转换 |
| array() | **返回数组** (值传递) |
| 复数 | real / imag() | 实/虚部 |
| conjugate() | **对应共轭复数** |
| 元素引用 | (r, c) / row(i) / col(i) | **指定**元素 |
| segment<n>(i) / block<nr, nc>(ir, ic) | **切片** |
| 运算 | << x0, x1, ... | 写入元素 |
| \* x | 矩阵乘法 |
| 变换 | transpose() | 派生转置 |
| asDiagnal() | 派生对角矩阵 |
| **reverse / reverseInPlace()** | **所有轴翻转** |
| 矩阵创建 | ::Identity(nr, nc) | 单位阵 |
| ::Zero / ::Ones / ::Random(nr, nc) | 默认值**填充** |
| ::Constant(nr, nc, v) / fill(v) | 指定值**填充** |
| normalize / normalized() | 单位化 **(原地/副本)** |
| 向量创建 | ::LinSpaced(size, low, high) | 等差**数列** |
| ::Unit(i) / ::UnitX / ::UnitY / ::UnitZ | 旋转向量 |
| 线性代数 | trace() | 迹 **(对角线之和)** |
| inverse() | 逆 **(include <.../Dense>)** |
| determinant() | 行列式 |
| eulerAngles(2, 1, 0) | **返回**欧拉角 |
| ldlt() | **返回**Cholesky分解 |
| jacobiSvd / bdcSvd(**DecompositionOptions** opt) | **返回**奇异值分解 |

|  |  |  |
| --- | --- | --- |
| Array<dtype, nr=-1, nc=-1> | | **数组** |
| 属性 | matrix() | **矩阵** (值传递) |
| 函数 | all / any / count / hasNaN() | 非零**判断** |
| mean / sum / prod / norm() | 均值 / 和 / 积 / 二范数 |
| sign / abs / sqrt / pow(x) | 基本 |
| maxCoeff / minCoeff() | **最值** |
| cwiseMax / cwiseMin(x) | 逐元素最值 |
| sin / cos / tan / asin / acos / atan() | **三角函数** |
| exp / log / log2 / log10 / logistic() | **指数函数** |
| round / ceil / floor() | 取整 |
| isInf / isFinite / isNaN() | 值**判断** |

## 稠密Dense：

|  |  |  |
| --- | --- | --- |
| **特征值分解：** | | |
| EigenSolver<mat>(mat) | | **普通分解器** |
| SelfAdjointEigenSolver<mat>(mat) | | **对称阵分解器** |
| 方法 | **eigenvalues()** | **特征值** |
| **eigenvectors()** | **特征向量** |

## 几何Geometry：

|  |  |  |
| --- | --- | --- |
| AngleAxisd(degree, axis) | | **旋转向量** |
| **运算** | \* vector | **对向量进行旋转** |
| **方法** | matrix() | **旋转矩阵** |

|  |  |  |
| --- | --- | --- |
| Quaterniond(w, x, y, z) | | **四元数 (旋转)** |
| **属性** | matrix() | **矩阵** |
| **创建** | ::Identity() | **单位旋转阵** |
| **方法** | coeffs() | **虚部、实部** |

|  |  |  |
| --- | --- | --- |
| Isometry3d::Identity() | | **等距变换 (齐次坐标系)** |
| **方法** | matrix() | **矩阵** |
| prerotate / rotate(rotation) | **原地旋转 (绝对/相对)** |
| pretranslate / translate(vec) | **原地平移 (绝对/相对)** |

## 奇异值SVD：

|  |  |  |
| --- | --- | --- |
| JacobiSVD<mat> | | **迭代分解** |
| BDCSVD<mat> | | **大矩阵分解** |
| **形参** | mat, **DecompositionOptions** opt | **分解模式** |
| **属性** | matrixU / singularValues / matrixV() |  |
| **方法** | solve(b) |  |

# 图优化g2o：

**package：G2O**

**lib：详见FindG2O.cmake**

**namespace：g2o**

**bin：g2o\_viewer \*.g2o**

## 内核core：

### 边base\_edge.h：

|  |  |  |
| --- | --- | --- |
| BaseEdge<dims, mea\_type> | | **边 (因变量，自动求导)** |
| BaseUnaryEdge<dims, mea\_type, vex\_type> | | **一元边** (base\_unary\_edge.h) |
| BaseBinaryEdge<dims, mea\_type, vex1\_type, vex2\_type> | | **二元边** (base\_binary\_edge.h) |
| BaseMultiEdge<dims, mea\_type> | | **多元边** (base\_multi\_edge.h) |
| 属性 | \_error | **误差向量** |
| \_measurement | **测量信息** |
| \_vertices | **顶点向量** |
| \_jacobianOplusXi | **雅可比矩阵** |
| chi2() | **残差平方和** |
| 抽象 | **bool** read(&is) | **加载** |
| **bool** write(&os) | **保存** |
| computeError() | **设置误差** |
| 虚拟 | linearizeOplus() | **设置雅可比矩阵 (非必须)** |
| 设置 | setId(i) | **序号** |
| setVertex(i, vertex) | **连接顶点** |
| setMeasurement(y) | **测量信息** |
| setInformation(eye) | **信息矩阵 [dims, dims]** |
| setLevel(x) | **分层优化层级** |
| setRobustKernel(\*k) | **鲁棒核** |

### 顶点base\_vertex.h：

|  |  |  |
| --- | --- | --- |
| BaseVertex<dims, type> | | **顶点** (自变量) |
| **属性** | \_estimate | **估计值** (通过 estimate() 访问) |
| **抽象** | **bool** read(&is) | **加载** |
| **bool** write(&os) | **保存** |
| setToOriginImpl() | **重置估计值** |
| oplusImpl(\*update) | **累加估计值** |
| **设置** | setMarginalized(true) | **待边缘化** |
| setFixed(true) | **定值** |
| setEstimate(et) | **估计值** |

### 求解block\_solver.h：

|  |  |
| --- | --- |
| BlockSolverPL<p, l>(linear\_solver) | **分块求解器** |

### 优化optimization\_algorithm\_\*.h：

|  |  |
| --- | --- |
| OptimizationAlgorithmGaussNewton(solver) | **高斯牛顿法** (optimization\_algorithm\_gauss\_newton.h) |
| OptimizationAlgorithmLevenberg(solver) | **LM算法** (optimization\_algorithm\_levenberg.h) |

### 鲁棒核robust\_kernel\_impl.h：

|  |  |
| --- | --- |
| RobustKernelHuber | **Huber核** |

### 图模型sparse\_optimizer.h：

|  |  |  |
| --- | --- | --- |
| SparseOptimizer | | **稀疏图** |
| 属性 | edges() | **边** |
| 设置 | setAlgorithm(algo) | **优化算法** |
| setVerbose(bool) | **输出信息** |
| setForceStopFlag(bool\*) | **停止信号** |
| addVertex(vex) | **添加顶点** |
| addEdge(edge) | **添加边** |
| 类型 | ::VertexIDMap | **顶点字典** |
| 启动 | **bool** initializeOptimization() | **初始化模型** |
| **int** optimize(iter) | **启动优化** |

## 求解solvers：

|  |  |
| --- | --- |
| type\_mat = BlockSolver::PoseMatrixType | |
| LinearSolverDense<type\_mat> | **稠密线性求解器 (**dense/linear\_solver\_dense.h**)** |
| LinearSolverEigen<type\_mat> | **Cholesky线性求解器 (**eigen/linear\_solver\_eigen.h**)** |

## 通用stuff：

### 运算misc：

|  |  |
| --- | --- |
| make\_unique<type>(...) | 初始化**类，返回**堆区指针 **(自动释放)** |

## 类型types：

### slam3d/types\_slam3d.h：

|  |  |  |
| --- | --- | --- |
| **顶点** | VertexSE3 | **SE3** |
| VertexPointXYZ | **XYZ** |
| **边** | EdgePointXYZ | **XYZ - XYZ** |
| EdgeSE3 | **SE3 - SE3** |
| EdgeSE3PointXYZ | **SE3 - XYZ (变换)** |
| EdgeSE3PointXYZDisparity | **SE3 - XYZ (视差)** |
| EdgeSE3PointXYZDepth | **SE3 - XYZ (深度)** |

|  |  |  |
| --- | --- | --- |
| SE3Quat(rota, pos) | | **实例化特殊欧氏群** |
| **属性** | matrix / matrix3x4() | **变换矩阵** |
| rotation() | **旋转四元数** |
| translation() | **平移向量** |
| **运算** | \* x | 矩阵乘法 |
| **函数** | log() / ::exp(vec) | **李代数 ↔ 李群** |
| ::hat(vec) / ::vee(mat) | **反对称矩阵 ↔ 向量** |
| inverse() | **求逆** |
| **构造** | toVector() / fromVector(v) | **向量转换** |

# 八叉树octomap/octomap.h：

**package：octomap**

**lib：${OCTOMAP\_LIBRARIES}**

**namespace：octomap**

**bin：octovis**

|  |  |
| --- | --- |
| point3d(x, y, z) | 3D点 |

|  |  |  |
| --- | --- | --- |
| Pointcloud | | 点云 |
| 方法 | push\_back(x, y, z) | **添加**点 |

|  |  |  |
| --- | --- | --- |
| OcTree | | **八叉树**地图 |
| 形参 | resolution | 分辨率 |
| file | 二进制文件 |
| 方法 | insertPointCloud(cloud, point3d origin) | **给定原点**插入点云 |
| updateInnerOccupancy() | 更新节点**占用** |
| writeBinary(file) | 导出**二进制文件** |

# 图像opencv2/opencv.hpp：

**C++：**

**package：OpenCV**

**lib：${OpenCV\_LIBS}**

**namespace：cv**

Python：复制Lib\site-packages\cv2\cv2.pyd到上一级

## 相机calib3d.hpp：

**点云x-y坐标的线性变换 (x-y轴缩放，原点平移，相机内参数)：**

|  |  |
| --- | --- |
| getOptimalNewCameraMatrix(cameraMatrix, distCoeffs,  imageSize, alpha, newImgSize) | 相机内参变换 |

|  |  |
| --- | --- |
| 标定： | |
| **bool** findChessboardCorners(img, chessSize, corners) | 寻找棋盘角点 |
| drawChessboardCorners(img, chessSize, corners) | 绘制棋盘角点 |
| cornerSubPix(gray, corners, winSize, zeroZone, criteria) | 修正角点位置 |
| calibrateCamera(objectPoints, imagePoints, imageSize, cameraMatrix,  distCoeffs, rvecs, tvecs, flags, criteria) | 相机标定 |

|  |  |  |
| --- | --- | --- |
| 去畸： | | |
| initUndistortRectifyMap(cameraMatrix, distCoeffs, R,  newCameraMatrix, size, m1type, mapx, mapx) | | 计算去畸坐标映射 |
| undistort(src, dst, cameraMatrix, distCoeffs) | | 图像去畸 |
| undistortPoints(src, dst, cameraMatrix, distCoeffs) | | 关键点去畸 |
| stereoRectify(cameraMatrix1, distCoeffs1, cameraMatrix2, distCoeffs2,  imgSize, R, T, R1, R2, P1, P2, Q) | | 双目矫正 |
| 形参 | Mat\_<float> cameraMatrix(3, 3); | 相机内参 |
| Mat\_<float> distCoeffs(5, 1); | 畸变参数 |

|  |  |  |
| --- | --- | --- |
| StereoSGBM | | 立体图像 |
| 方法 | ::create(0, 96, 9, 8\*9\*9, 32\*9\*9, 1, 63, 10, 100, 32) | 初始化 |
| compute(left, right, dst) | 计算视差 (CV\_16S) |

**// 齐次坐标 (x, y, z, w) → 非齐次坐标 (x/w, y/w, z/w)**

|  |  |
| --- | --- |
| 2D-2D对极几何： | |
| findFundamentalMat(pts1, pts2, method) | 计算基本矩阵F |
| findEssentialMat(pts1, pts2, cameraMatrix) | 计算本质矩阵E () |
| findHomography(pts1, pts2, method) | 计算单应矩阵H (纯旋转) |
| recoverPose(E, pts1, pts2, cameraMatrix, R, t) | 恢复三维变换信息 (单位向量t) |
| triangulatePoints(projMatr1, projMatr2, pts1, pts2, points4D) | 三维点重建，返回点云齐次坐标 |

|  |  |
| --- | --- |
| 旋转： | |
| Rodrigues(src, dst) | 旋转向量 ↔ 旋转矩阵 |

|  |  |
| --- | --- |
| 3D-2D PnP： | |
| solvePnP(objPts1, imgPts2, cameraMatrix, distCoeffs, R, t) | 计算三维变换信息 |

## 内核core.hpp：

|  |  |  |
| --- | --- | --- |
| FileStorage(file, FileStorage::Mode flags, encoding) | | xml / yaml / json文件  yaml头：%YAML:1.0 |
| 属性 | isOpened() | 打开 |
| FileNode operator[](key) | 键值查找 |
| 方法 | release() | 释放 |

|  |  |  |
| --- | --- | --- |
| **RNG(seed)** | | **随机数生成器** |
| **方法** | **uniform(low, high)** | **均匀分布** |
| **gaussion(sigma)** | **正态分布** |

|  |  |
| --- | --- |
| Ptr<dtype> | 引用计数指针 |
| TermCriteria(type, maxCount, eps) | **迭代**控制 |
| parallel\_for\_(**Range** range, functor) | 并行运算 |

|  |  |
| --- | --- |
| Size(w, h) | 2D尺寸 |
| Point(x, y) | 2D点 |
| Point3d(x, y, z) | 3D点 |
| Scalar(x, y, z, w) | 像素值 |

|  |  |  |
| --- | --- | --- |
| Rect | | 2D切片 |
| 形参 | x, y, w, h | 边界框 |
| pt1, pt2 | 对角线 |
| 方法 | tl / br() | 边界点 |
| area() | 面积 |
| contains(pt) | 判断包含点 |

|  |  |  |
| --- | --- | --- |
| Mat | | 图像 |
| Mat\_<dtype> | | 矩阵 |
| 形参 | rows, cols, type, s | 2D矩阵 |
| ndims, sizes, type, s | nD矩阵 |
| 运算 | << x0, x1, ... | 写入元素 |
| 属性 | rows / cols / channels() | 形状 |
| dims / size() | 维度 |
| total() | 元素数 **(需乘通道)** |
| elemSize() | **元素**占用空间 |
| type / depth() | 类型 (位于**core/hal/interface.h**) |
| empty() | 为空 |
| 访问 | ptr<T>(0) / ptr<T>(r, c) | 数据 **/** 像素头指针 |
| at<T>(idx) | 像素 |
| row / col(i) | 单行 |
| rowRange / colRange(s, e) | 多行 |
| (rect) | 任意矩形 |
| 拷贝 | clone() / copyTo(mat) | 深拷贝 |
| convertTo(dst, rtype, w, b) | 类型转换**，并线性变换** |
| 填充 | ::eye | 单位阵，形参同初始化 |
| ::zeros / ::ones | 默认值，形参同初始化 |
| setTo(value, mask) | 指定值 |

|  |  |  |
| --- | --- | --- |
| 2D操作： | | |
| 常规操作 | print(src) | 输出 |
| flip(src, dst, flipCode) | 翻转 |
| 数值计算 | repeat(src, ny, nx) | 重复 |
| LUT(src, lut, dst) | 查表取值 |
| 形状修改 | copyMakeBorder(src, dst, t, b, l, r, borderTypes btype, value) | 边界填充 |
| hconcat / vconcat(std::vector<Mat> src, dst) | 图像拼接 |
| transpose(src, dst) | 转置 |
| reshape(cn, rows) | 重塑 |

|  |  |
| --- | --- |
| 内存管理： | |
| copyTo(src, dst, mask) | 拷贝 |
| convertFp16(src, dst) | CV\_32F ↔ CV\_16S |

|  |  |
| --- | --- |
| 数值计算： | |
| add / subtract / multiply / divide /  bitwise\_and / bitwise\_or / bitwise\_xor(src1, src2, dst) | 逐元素运算 (可用运算符) |
| addWeighted(src1, alpha, src2, beta, gamma) | 逐元素加权 |
| max / min(src1, src2, dst) | 逐元素最值 |
| sum / mean(src) | 逐通道聚合 |
| sqrt(src, dst) / pow(src, power, dst) | 幂运算 |
| exp / log(src, dst) | 对数运算 |
| polarToCart / cartToPolar(x, y, magnitude, angle) | 极坐标转换 |
| norm(src) | 聚合 |
| hasNonZero / countNonZero(src) | 非零聚合 |
| flipND(src, dst, axis) | 翻转 |
| normalize(src, dst, alpha, beta, norm\_type) | 规范化 (默认单位化) |
| minMaxLoc(src, minVal, maxVal, minLoc, maxLoc) | 最值及位置 |

|  |  |
| --- | --- |
| 形状修改： | |
| merge(std::vector<Mat> src, dst) | 通道拼接 |
| split(src, std::vector<Mat> dst) | 通道拆分 |
| rotate(src, dst, RotateFlags rotateCode) | 旋转 |
| transposeND(src, std::vector<int> order, dst) | 通道重排 |
| reshape(nc, std::vector<int> newshape) | 重塑 |

## 特征feature2d.hpp：

|  |  |  |
| --- | --- | --- |
| **Keypoint** | | **关键点** |
| **属性** | **pt** | **位置** |
| **size** | **邻域直径** |
| **angle** | **夹角** |
| **response** | **响应** |
| **octave** | **金字塔层** |
| **方法** | **::convert(std::vector<Keypoint>, std::vector<Point2f>)** | **转换** |

|  |  |  |
| --- | --- | --- |
| **DMatch** | | **关键点描述子匹配** |
| **属性** | **queryIdx / trainIdx** | **关键点索引** |
| **distance** | **距离 (< 运算符)** |

|  |  |
| --- | --- |
| **绘制：** | |
| drawKeypoints(image, keypoints, outImage, **DrawMatchesFlags** flags) | 关键点 |
| drawMatches(img1, img2, matches1to2, outImg) | 匹配结果 |

|  |  |  |
| --- | --- | --- |
| **GFTT特征：** | | |
| GFTTDetector | | **GFTT特征**提取器 |
| **方法** | ::create(maxCorners, qualityLevel, minDistance) | 初始化 |
| detect(image, keypoints) | **检测**关键点 |

|  |  |  |  |
| --- | --- | --- | --- |
| **ORB特征：** | | | |
| **代码** | ORB | | **ORB特征**提取器 |
| **方法** | ::create(nfeatures) | 初始化 |
| detect(image, keypoints) | **检测**关键点 |
| compute(image, keypoints, descriptors) | **计算**描述子 |
| detectAndCompute(image, mask, keypoints, descriptors) | **检测并计算** |
| **Oriented FAST**  **关键点** | 角点 | **邻域圆的16个像素、中心点像素间比较，筛选角点** | |
| 尺度 | 图像金字塔分层检测 | |
| 旋转 | 中心点→质心 (**sum(灰度 \* 坐标) / sum(灰度)**) | |
| BRIEF描述子 | 随机选点比较得到0/1，转存为uint8 (使用汉明距离匹配) | | |

|  |  |  |
| --- | --- | --- |
| DescriptorMatcher | | 描述子匹配器 |
| **方法** | ::create("BruteForce-Hamming") | 初始化 |
| match(queryD, trainD, std::**vector<DMatch>** matches) | 匹配 |

**// 以max(min\_DMatch->distance, 30.0) 为阈值，筛选匹配结果**

## 交互highgui.hpp：

|  |  |  |
| --- | --- | --- |
| imshow(winname, mat) | 显示图像 | |
| waitKey(delay) | 0 | 无限期等待键盘输入 |
| int | 等待对应ms后关闭 |
| destroyAllWindows() | 关闭所有展示窗口 | |

## 编码imgcodecs.hpp：

|  |  |  |
| --- | --- | --- |
| **图像编码：** | | |
| **.png** | **CV\_8U / CV\_16U** | **无损压缩，支持BGRA** |
| **.jpg / .jpeg** | **CV\_8U / CV\_16U** | **有损压缩** |
| **.tiff** | **CV\_8U / CV\_16U / CV\_32F / CV\_64F** | **支持多图像** |
| **.exr** | **CV\_32F** | **支持压缩** |
| **.hdr** | **CV\_32F** | **无压缩** |

|  |  |
| --- | --- |
| imread(filename, ImreadModes flags = -1) | 打开图像返回BGR图像 |
| imwrite(filename, img) | 导出图像到文件 |

## 运算imgproc.hpp：

四通道：Blue, Green, Red, Opacity

Gray = 0.299 \* R + 0.587 \* G + 0.114 \* B

|  |  |
| --- | --- |
|  | |
| 色调 H |  |
| 饱和 S |  |
| 亮度 V |  |

|  |  |
| --- | --- |
| resize(src, dst, size) | 图像重置尺寸 |
| cvtColor(src, dst, ColorConversionCodes code) | 颜色空间转换 |
| threshold(src, dst, thresh, maxval, ThresholdTypes type) | 二值化 |
| remap(src, dst, mapx, mapy, InterpolationFlags flags) | 像素坐标重映射 |

|  |  |  |
| --- | --- | --- |
| 滤波： | | |
| filter2D(src, dst, ddepth, kernel) | 单通道卷积 | |
| blur(src, dst, ksize) | 均值滤波 | |
| GaussianBlur(src, dst, ksize, sigmaX=1) | 高斯滤波 | |  |  |  | | --- | --- | --- | | 0.6 | 0.8 | 0.6 | | 0.8 | 1 | 0.8 | | 0.6 | 0.8 | 0.6 | |
| medianBlur(src, dst, ksize) | 中值滤波 | |

|  |  |  |  |
| --- | --- | --- | --- |
| 腐蚀膨胀： | | | |
| erode(src, dst, kernel, iterations, borderValue) | | 腐蚀 (最小池化) | |
| dilate(src, dst, kernel, iterations, borderValue) | | 膨胀 (最大池化) | |
| morphologyEx(src, dst, MorphTypes op,  kernel, iterations, borderValue) | | 形态学运算 | |
| op：  运算模式 | MORPH\_OPEN | 开运算除去毛刺 | 腐蚀 => 膨胀 |
| MORPH\_CLOSE | 闭运算 | 膨胀 => 腐蚀 |
| MORPH\_GRADIENT | 梯度运算求边缘 | 膨胀 - 腐蚀 |
| MORPH\_TOPHAT | 礼帽求毛刺 | 原图 - 开运算 |
| MORPH\_BLACKHAT | 黑帽 | 闭运算 - 原图 |

|  |  |  |
| --- | --- | --- |
| 梯度算子： | | |
| Sobel(src, dst, ddepth, dx, dy, ksize) | Sobel | |  |  |  | | --- | --- | --- | | -1 | 0 | +1 | | -2 | 0 | +2 | | -1 | 0 | +1 | |
| Scharr(src, dst, ddepth, dx, dy) | Scharr | |  |  |  | | --- | --- | --- | | -3 | 0 | +3 | | -10 | 0 | +10 | | -3 | 0 | +3 | |
| Laplacian(src, dst, ddepth, ksize) | Laplacian | |  |  |  | | --- | --- | --- | | 0 | 1 | 0 | | 1 | -4 | 1 | | 0 | 1 | 0 | |

|  |  |  |
| --- | --- | --- |
| 直方图均衡： | | |
| Ptr<CLAHE> createCLAHE(clipLimit, tileGridSize) | | CLAHE算法 |
| 方法 | apply(src, dst) | 应用 |

|  |  |  |  |
| --- | --- | --- | --- |
| 轮廓检测： | | | |
| findContours(gray, contours, RetrievalModes mode, ContourApproximationModes method) | | | |
| mode：  检索模式 | RETR\_EXTERNAL | 外轮廓 | |
| RETR\_LIST | 所有  轮廓 | 保存到列表 |
| RETR\_CCOMP | 分为外轮廓、内轮廓 |
| RETR\_TREE | 重构嵌套轮廓的整个层次 |
| method：  逼近方法 | CHAIN\_APPROX\_NONE | 以Freeman链码的方式输出轮廓，  所有其他方法输出多边形 (顶点的序列) | |
| CHAIN\_APPROX\_SIMPLE | 压缩直线部分，只保留终点部分 | |

|  |  |  |
| --- | --- | --- |
| 曲线： | | |
| contourArea(contour) | 返回曲线属性 | 面积 |
| arcLength(curve, closed=False) | 周长 |
| boundingRect(curve\_or\_gray) | 曲线/图像外接矩形信息 | |
| minEnclosingCircle(points, center, radius) | 外接圆信息 | |

|  |  |
| --- | --- |
| 绘制： | |
| line(img, pt1, pt2, color, thickness) | 直线 |
| rectangle(img, pt1, pt2, color, thickness) | 矩形 |
| circle(img, center, radius, color, thickness) | 圆形 (实心：thickness=-1) |
| ellipse(img, center, axes, angle,  startAngle, endAngle, color, thickness) | 椭圆 |
| polylines(img, pts, isClosed, color, thickness) | 多边形 |
| putText(img, text, org, fontFace, fontScale, color, thickness) | 字符串 |
| drawContours(img, contours, idx, color, thickness) | 轮廓 (idx：索引，-1所有) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模板匹配： | | | | | |
| matchTemplate(image, templ, result, TemplateMatchModes method) | | | | | |
| method：  计算方法 | TM\_SQDIFF\_NORMED | 归一化 | 平方差 | 越 → 0 | 越相关 |
| TM\_CCORR\_NORMED | 相关性 | 越 → 1 |
| TM\_CCOEFF\_NORMED | 相关系数 |

## 视频video.hpp：

|  |  |  |
| --- | --- | --- |
| VideoCapture(file=0) | | 打开视频 / 摄像头 |
| 运算 | >> img | 读取视频帧 |
| 方法 | isOpened() | 检测状态 |
| read(dst) | 读取视频帧 |
| get(VideoCaptureProperties propID) | 获取参数 |
| set(VideoCaptureProperties propID, value) | 设置参数 |

|  |  |
| --- | --- |
| VideoCaptureProperties | |
| CAP\_PROP\_FRAME\_WIDTH | 图像宽度 |
| CAP\_PROP\_FRAME\_HEIGHT | 图像高度 |
| CAP\_PROP\_FRAME\_COUNT | 帧数 |
| CAP\_PROP\_FPS | FPS |
| CAP\_PROP\_POS\_MSEC | 视频当前位置 (ms) |
| CAP\_PROP\_CONTRAST | 相机亮度 |
| CAP\_PROP\_GAIN | 相机白平衡 |
| CAP\_PROP\_EXPOSURE | 相机曝光 |

|  |  |  |
| --- | --- | --- |
| 光流法： | | |
| 代码 | calcOpticalFlowPyrLK(prevImg, nextImg, prevPts,  nextPts, status, err) | LK稀疏光流 (关键点跟踪) |
| calcOpticalFlowFarneback(prev, next, flow, pyr\_scale,  levels, winsize, iterations, poly\_n, poly\_sigma, flags) | Farneback稠密光流 (流图像) |
| 假设 | 连续两帧图像之间，目标的像素亮度不改变；相邻的像素之间有相似的运动 | |
| 意义 | 像素的位移值，两通道分别表示x,y | |

# 点云pcl：

**package：PCL**

**lib：${PCL\_LIBRARIES}**

**namespace：pcl**

**bin：pcl\_viewer \*.pcd**

## 滤波filters：

|  |  |  |
| --- | --- | --- |
| Filter<PointT> | | **滤波器**基类 **(filter.h)** |
| **设置** | setInputCloud(pcl) | 输入点云 |
| **方法** | filter(pcl) | 输出**滤波结果** |

|  |  |  |
| --- | --- | --- |
| StatisticalOutlierRemoval<PointT> | | **基于点邻域统计的**离群点滤除  **(statistical\_outlier\_removal.h)** |
| **设置** | setMeanK(k) | **平均距离估计的**点数 |
| setStddevMulThresh(stddev\_mult) | 标准差倍数 |

|  |  |  |
| --- | --- | --- |
| VoxelGrid<PointT> | | 3D体素网格**过滤器，保留每个体素的质心**  **(voxel\_grid.h)** |
| **设置** | setLeafSize(lx, ly, lz) | 体素尺寸 |

## 流io：

namespace：io

|  |  |
| --- | --- |
| **pcd\_io.h：** | |
| **loadPCDFile("\*.pcd", pcl)** | 加载点云**文件** |
| savePCDFileBinary("\*.pcd", pcl) | 导出点云**文件** |

## 点云point\_cloud.h：

|  |  |  |
| --- | --- | --- |
| PointCloud<PointT> | | 点云 |
| **形参** | pc, indices | **深拷贝** |
| **运算** | + pcl | **叠加点云** |
| **属性** | ::Ptr | **引用计数指针** |
| is\_dense | **点无缺失** |
| points | **点容器** |
| size() | **大小** |

## 点类point\_types.h：

|  |  |  |
| --- | --- | --- |
| PointXYZRGB(r, g, b) | | RGB**空间点** |
| PointXYZRGBNormal | | RGB**空间点 (带**法线**)** |
| **属性** | r / g / b / x / y / z | 颜色 **/** 位置 |
| normal\_x / normal\_y / normal\_z | 法线 |

## 网格PolygonMesh.h：

|  |  |
| --- | --- |
| PolygonMesh | 三角网格 |

## 搜索search：

namespace：search

|  |  |  |
| --- | --- | --- |
| KdTree<PointT> | | k-d树 **(kdtree.h)** |
| **设置** | setInputCloud(pcl) | 输入点云 |

## 表面surface：

|  |  |  |
| --- | --- | --- |
| MovingLeastSquares<PointInT, PointOutT> | | 移动最小二乘 **(mls.h)** |
| **设置** | setInputCloud(pcl) | 输入点云 |
| setSqrGaussParam(radius\_square) | 高斯参数**的平方** |
| setComputeNormals(bool) | **启用**法线**计算** |
| setSearchMethod(tree) | 搜索算法 |
| setSearchRadius(radius) | 搜索半径 |
| setPolynomialFit(bool) | **启用**多项式近似 |
| setPolynomialOrder(order) | **多项式拟合**阶数 |
| **方法** | process(pcl) | 输出**表面平滑结果** |

|  |  |  |
| --- | --- | --- |
| GreedyProjectionTriangulation<PointT> | | 贪婪三角测量 **(gp3.h)** |
| **设置** | setInputCloud(pcl) | 输入点云 |
| setMu(2.5) | 平滑度 |
| setNormalConsistency(bool) | **保持**法线一致 |
| setSearchMethod(tree) | 搜索算法 |
| setSearchRadius(radius) | 三角边长 |
| setMaximumNearestNeighbors(nnn) | 近邻数量 |
| setMaximumSurfaceAngle(x) | 法线夹角**限制** |
| setMaximumAngle / setMinimumAngle(x) | 三角内角**限制** |
| **方法** | reconstruct(mesh) | 输出**表面重建结果** |

## 可视化visualization：

namespace：visualization

|  |  |  |
| --- | --- | --- |
| PCLVisualizer | | 点云可视化 **(pcl\_visualizer.h)** |
| **方法** | addPolylineFromPolygonMesh(mesh, id) | **添加**多边形 |
| addPolygonMesh(mesh, id) | **添加**多边形网格 |
| resetCamera() | 重置**相机** |
| spin() | **调用**交互**器，进入事件循环** |

# 李群sophus：

**package：Sophus**

**lib：Sophus::Sophus**

**namespace：Sophus**

|  |  |  |
| --- | --- | --- |
| 群 (集合 + 运算) | **封闭性** |  |
| **结合律** |  |
| **幺元** |  |
| **逆** |  |

|  |  |  |
| --- | --- | --- |
| **李群** | **定义** | **具有连续性质的群** |
| **李代数** | **定义** | **单位元附近的正切空间** |
| **封闭性** |  |
| 双线性 |  |
| **自反性** |  |
| **雅可比等价** |  |

****

扰动模型：全0李代数，添加扰动量1e-4后，转换为李群，左乘于李群

欧拉角 (rpy角)：旋转顺序ZYX，万向锁问题 (奇异性问题 - 绕y旋转后，x轴、原z轴共线)

## 正交so3.hpp：

|  |  |  |
| --- | --- | --- |
| SO3d(rota) | | **实例化特殊正交群 (旋转矩阵 / 四元数)** |
| **属性** | matrix() | **旋转矩阵** |
| **运算** | \* x | 矩阵乘法 |
| **函数** | log() / ::exp(vec) | **李代数 ↔ 李群** |
| ::hat(vec) / ::vee(mat) | **反对称矩阵 ↔ 向量** |

## 欧氏se3.hpp：

|  |  |  |
| --- | --- | --- |
| SE3d(rota, pos) | | **实例化特殊欧氏群 (旋转矩阵 / 四元数)** |
| **属性** | matrix / matrix3x4() | **变换矩阵** |
| unit\_quaternion() | **旋转四元数** |
| translation() | **平移向量** |
| **运算** | \* x | 矩阵乘法 |
| **函数** | log() / ::exp(vec) | **李代数 ↔ 李群** |
| ::hat(vec) / ::vee(mat) | **反对称矩阵 ↔ 向量** |