Midterm

pclStatisticalOutlierRemoval

- pointCloud: pcl::PointXYZ
- ktree: int
- standDevMulThresh: double
- cloudPtr: pcl::PointXYZ::ptr
- sor: pcl::statisticalOutlierRemoval

+ create():StatisticalOutlierRemoval
+ setInputCloud(&PointXYZ)
+ getInputCloud(&PointXYZ)
+ setMeanK(int)
+ getMeanK():int
+ setStddevMulThresh(double)
+ getStddevMulThresh():double
+ setfilter(&PointXYZ)

pclIo

+ pointCloud: PointXYZ

+ pclIo()
+ readPCDfile(&string)
+ getPointCloud(&PointXYZ)

pclCloudViewer

- user_data: int

+ cloudViewer()
+ display(&PolygonMesh)

pclMlsSmoothing

- searchRadius: double
- pointCloud: pcl::PointXYZ
- cloudPtr: pcl::PointXYZ::ptr

+ pclMlsSmoothing()
+ setSearchRadius(double)
+ getSearchRadius(): double
+ setInputCloud(&PointXYZ)
+ getInputCloud(&PointXYZ)
+ mlsProcess(&PointNormal)

pclPassThrough

- pointCloud: pcl::PointXYZ
- zMin: double
- zMax: double
- yMin: double
- yMax: double
- xMin: double
- xMax: double
- cloudPtr: pcl::PointXYZ::ptr
- pass: pcl::passThrough

+ passThrough()
+ setInputCloud(&PointXYZ)
+ getInputCloud(&PointXYZ)
+ setFilterZlimit(float&,float&)
+ setFilterYlimit(float&,float&)
+ setFilterXlimit(float&,float&)
+ getFilterLimit: vector<float>
+ filterProcess(&PointXYZ)

pclVoxel

- lx: float
- ly: float
- lz: float
- pointCloud: pcl::PointXYZ
- cloudPtr: pcl::PointXYZ::ptr

+ voxel()
+ setLeafSize(float, float, float)
+ getLeaflxSize(): vector<float>
+ setInputCloud(&PointXYZ)
+ getInputCloud(&PointXYZ):
+ filterProcess(&PointXYZ)

pclFastTriangular

- cloud: pcl::PointNormal
- cloudPtr: pcl::PointNormal::ptr
- searchRadius: double
- gp3:GreedyProjectionTriangulation<
  pcl::PointNormal>

+ pclFastTriangular()
+ setSearchRadius(double)
+ getSearchRadius(): double
+ setInputCloud(&PointNormal)
+ getInputCloud(&PointNormal)
+ reconstruct(&PolygonMesh)
+ getSegID():vector<int>

Final

findNearestPoint

- specificPoints:vector<double>
- cloud: pcl::PointNormal

+readtext(&String):vector<double>
+setInputCloud(&PointNormal)
+findNearestProcess():vector<int>
+setPosition(vector<float>& pos): void

kukaControl

+ kukaControl()
+ initializeJoints(KDL::JntArray)
+ initializePoints(JointTrajectoryPoint,int, float)
+ nameJoints(JointTrajectory, int)
+ evalPoints(JointTrajectoryPoint, KDL::JntArray ,int)

delaunay3

- pts: vector<Shx>
- pt: Shx
- cloud: pcl::PointNormal

+ delaunay3()
+ setInputCloud(&PointNormal)
+ putPointCloudIntoShx()
+ processDelaunay(vector<Triad>)
+ getShx(vector<Shx>)

dijkstraPQ

- V: int
- parent: int*
- adj: list<pair<int, double>>*
- triads: vector<Triad>
- triPartID: vector<int>
- cloud: pcl::PointNormal
- cloudPtr: pcl::PointNormal::ptr

+ dijkstraPQ()
+ addEdge(int,int,double)
+ shortestPath(int,int)
+ cal2Point(int ,int):double
+ distanceCalculation(std::vector<int>&)
+ setTri(vector<Triad>)
+ computeWeight()
+ returnDijkstraPath(int, int, vector<int>)
+ setInputCloud(&PointNormal)
+ returnDijkstraPath(int, int, vector<position>)