# Memoria

Estructura de capítulos:

1. Capítulo introductorio: Fundamento teórico
2. Capítulo 2: PCL a nivel técnico (formato, filter, SIFT y registration), objetivos, herramientas
3. Capítulo 3: visualización, extracción SIFT
4. Capítulo 4: Extracción de normales
5. Capítulo 5: Extracción de SIFT
6. Capítulo 6: Optimización
7. Capítulo 7: Resultados

# Diagramas

Tengo que hacer diagramas de:

* Visualization
  + Alto nivel
  + Bajo nivel o explicar el código a cachos
* Visualization\_sift
  + Alto nivel
  + Bajo nivel o explicar el código a cachos
* Programa principal de extracción de SIFT points
  + Alto nivel
  + Bajo nivel o explicar el código a cachos
* Parte de extracción de normales
  + Setinputcloud
  + Setradiussearch
  + Setsearchmethod
  + Compute
* Parte de extraer SIFT
  + …

# Medición de tiempos

Haciendo uso de:

#include <iostream>

#include <ctime>

clock\_t begin,end;

double elapsed\_sec = double (end-begin)/CLOCKS\_PER\_SEC;

Tengo tiempos medidos con precisión del programa principal:

* Tiempo para cargar nube
* Tiempo para extraer normales
* Tiempo para copiar xyz info de la nube de entrada en la nube de normales
* Tiempo de extracción de SIFT

Metiendo mano en mido tiempos de forma precisa según el diagrama de la parte de normal estimation:

## feature.hpp

* **pcl::feature compute** y se miden tiempos de:
  + **pcl::Feature initCompute + pcl::base initCompute**
  + copiar info de nube de entrada en la de salida
  + **pcl::NormalEstimation computeFeature**
* **pcl::Feature initCompute**
* **pcl::Feature deinitCompute**
* **pcl::Feature solvePlaneParameters**

## feature.h

* **pcl::feature searchForNeighbors**

## normal\_3d.hpp

* Aquí está el método **pcl::NormalEstimation computeFeature** y mido tiempos de:
  + searchForNeighbors
  + computePointNormal
    - computeMeanAndCovarianceMatrix
    - solvePlaneParameters
  + flipNormalTowardsViepoint

Para medir tiempos de computeMeanAndCovarianceMatrix y flipNormalTowardsViewpoint se crean e inicializan las variables time\_covariance y time\_solvePlaneParameters que se pasan a computePointNormal

## normal\_3d.h

* Aquí están las implementaciones del método **pcl::NormalEstimation computePointNormal** que reciben dos argumentos adicionales

*double &time\_covariance*

*double &time\_solvePlaneParameters*

En los cuales se acumulan los tiempos de calcular la matriz de covarianzas y sacar autovalores, autovectores y curvatura (línea 242)

* También está el método **pcl::NormalEstimation flipNormalTowardsViewpoint** dentro del cual no se mide nada

## pcl\_base.hpp

* **pcl::PCLBase initCompute**

## centroid.hpp

* **pcl::computeMeanAndCovarianceMatrix**

# HLS

yoqse tio xdd