NDT registration algorithms

using point clouds

```
#include <ndt_gslam/registration/d2d_ndt2d.h>

int main(int argc, char ** argv)
{
    typedef pcl::PointCloud<pcl::PointXYZ> Pcl;
    Pcl:Ptr target(new Pcl());
    Pcl:Ptr source(new Pcl());
    Pcl out_pcl;

    pcl::D2DNormalDistributionsTransform2D<pcl::PointXYZ, pcl::PointXYZ> matcher;
    matcher.setInputSource(source);
    matcher.setInputTarget(target);
    matcher.align(out_pcl);
    matcher.getFinalTransformation();
}
```

using NDT frames

```
#include <ndt_gslam/registration/d2d_ndt2d.h>
#include <ndt_gslam/ndt/ndt_grid2d.h>
#include <ndt_gslam/ndt/ndt_cell.h>

int main(int argc, char ** argv)
{
    typedef slamuk::NDTGrid2D<slamuk::NDTCell, pcl::PointXYZ> NDTFrame;
    NDTFrame::Ptr target(new NDTFrame());
    NDTFrame::Ptr source(new NDTFrame());
    pcl::PointCloud<pcl::PointXYZ> out_pcl;

    pcl::D2DNormalDistributionsTransform2D<pcl::PointXYZ, pcl::PointXYZ> matcher;
    matcher.setInputSource(source);
    matcher.setInputTarget(target);
    matcher.align(out_pcl);
    matcher.getFinalTransformation();
}
```

The same API can be used for other registration algorithms in our package.

Define a new NDT cell

In order to define new NDT cell, cell class needs to implement following functions

```
typedef typename Eigen::Vector3d Vector;
typedef typename Eigen::Matrix3d Matrix;
typedef Eigen::Transform<double, 3, Eigen::TransformTraits::Affine> Transform;


bool hasGaussian() const; // true if cell includes gaussian
const Vector &getMean() const;
const Matrix &getCov() const;
const Matrix &getICov() const;
int8_t getOccupancy() const;
void addPoint(const Vector &pt);
void computeGaussian();
void updateOccupancy(const Vector &start, const Vector &end, size_t new_points); // update b
NDTCell &operator+=(const NDTCell &other);
void transform(const Transform &trans);
```

Define new type data holder for the graph slam interface

The Graph slam interface expect one template parameter. This parameter represents environment measurement data. This data holder will be saved inside of the pose graph nodes. From here it will be used for registration of loop closures. Development of own data holder is strongly related with implementation of the loop closure registration. Data stored in this data wrapper must be accepteble by scan matcher after calling getData() e.g. setInputSource(a.getData()); New data type for the graph slam interface needs to include these methods:

```
Eigen::Vector2d getCentroid() const;  // used in loop detection
double getRadius() const; // used in loop detection
const Data &getData() const;  // used for registration
void updatePosition(const Eigen::Vector3d &new_pose); // used for update of the measurement
// data when node changes its possition.
```

The Data holders for NDT grid and the Point Cloud are provided in the package.

Voxel grid cell type

A cell type for voxel grid needs to implement operator+= and operator=. Our implementation of the voxel grid is efficient when using bigger datatypes as a cell.