栅格地图算法流程(持续更新)

一.定义需要的类

1.参数

1.1 用于叠加动态和静态的栅格之后的地图类的构造函数

```
struct Params
{
    包含创建栅格地图需要的参数的尺寸,分辨率,占用的阈值等;
};
```

1.2 用于静态栅格地图类的构造函数

```
struct StaticParams : public Params
{
    静态栅格地图需要的参数,增加占用率更新相关参数;
};
```

1.3 用于动态栅格地图类的构造函数

```
struct DynamicParams : public Params
{
    动态栅格需要的参数,增加粒子的相关参数;
}
```

2.地图类

2.1 静态栅格地图类

```
Class StaticGridMap
{
public:
    StaticGridMap(StaticParams& params);
    ~StaticGridMap();
    void initialize_map(); //用第一帧信息初始化静态地图
```

```
    void update_measurement(); //更新观测栅格数组
    void update_pose(); //更新位置
    void update_map(); //更新静态栅格地图(占用情况,位置变化等)
    vector<StaticMapCell*> get_cell();//获取栅格地图数组
private:
    StaticParams static_params_;//静态栅格地图参数
    vector<StaticMapCell*> static_map_cell; //静态栅格地图
    vector<StaticMeasurementCell*> static_measurement_cell;//用于更新的每一帧的观测值
    Position position_;  //位置信息,用于更新栅格地图标号(用于记录里程计信息,不一定需要)
};
```

2.2 动态栅格地图类

```
Class DynamicGridMap
{
public:
     DynamicGridMap(DynamicParams& params);
    ~DynamicGridMap();
    void initialize_map(); //用第一帧信息初始化动态地图
    void update_update_measurement(); //更新观测栅格数组
    void update_pose(); //更新位置
    vector<DynamicMapCell*> get_cell();//获取栅格地图数组
    //通过更新粒子更新栅格状态
    void initializeParticles();
    void particlePrediction(float dt);
    void particleAssignment();
    void gridCellOccupancyUpdate();
    void updatePersistentParticles();
    void initializeNewParticles();
    void statisticalMoments();
    void resampling();
private:
    DynamicParams dynamic_params_; //动态栅格地图参数
    vector<DynamicMapCell*> Dynamic_map_cell_; //动态栅格地图
    vector<DynamicMeasurementCell*> Dynamic_measurement_cell_;//用于更新的观测值
```

```
    ParticlesSoA particle_array_;
    ParticlesSoA particle_array_next_;
    ParticlesSoA birth_particle_array_;
    Position position_;  //位置信息,用于更新栅格地图标号
};
```

2.3 静态和动态叠加后的栅格地图类(给下游规划的栅格地图类)

```
Class GridMap
{
    Class GridMapCell
    {
    public:
        float get_v_x();//获取x方向速度
        float get_v_y();//获取y方向速度
        Point2Type get_center_point(); //获取中心点坐标
        Classification get_classfication(); //获取网格类别
        bool is_oncoming();//是否为对向来车
        bool is_occupied();//是否被占据(在参数中设定阈值)
    public:
      Point2Type  center_point_; //中心点坐标
      float velocity_x_;//沿x轴速度
      float velocity_y_;//沿y轴速度
      Classification classfication_;//障碍物类别
      bool occupy_;//是否被占用
    }
public:
    GridMap(Params& params);
    ~GridMap();
    int get_height(); //获取地图高度
    Int get_width();//获取地图宽度
    float get_resolution();//获取地图分辨率
    Timestamp get_timestamp();//获取时间戳
    vector< GridMapCell* > get_cell();//获取栅格地图数组
private:
Params params_;//地图参数
```

```
    Timestamp timestamp_;//地图时间戳
    vector<GridMapCell*>  grid_map_cell; //栅格地图
    };
```

3.栅格结构体

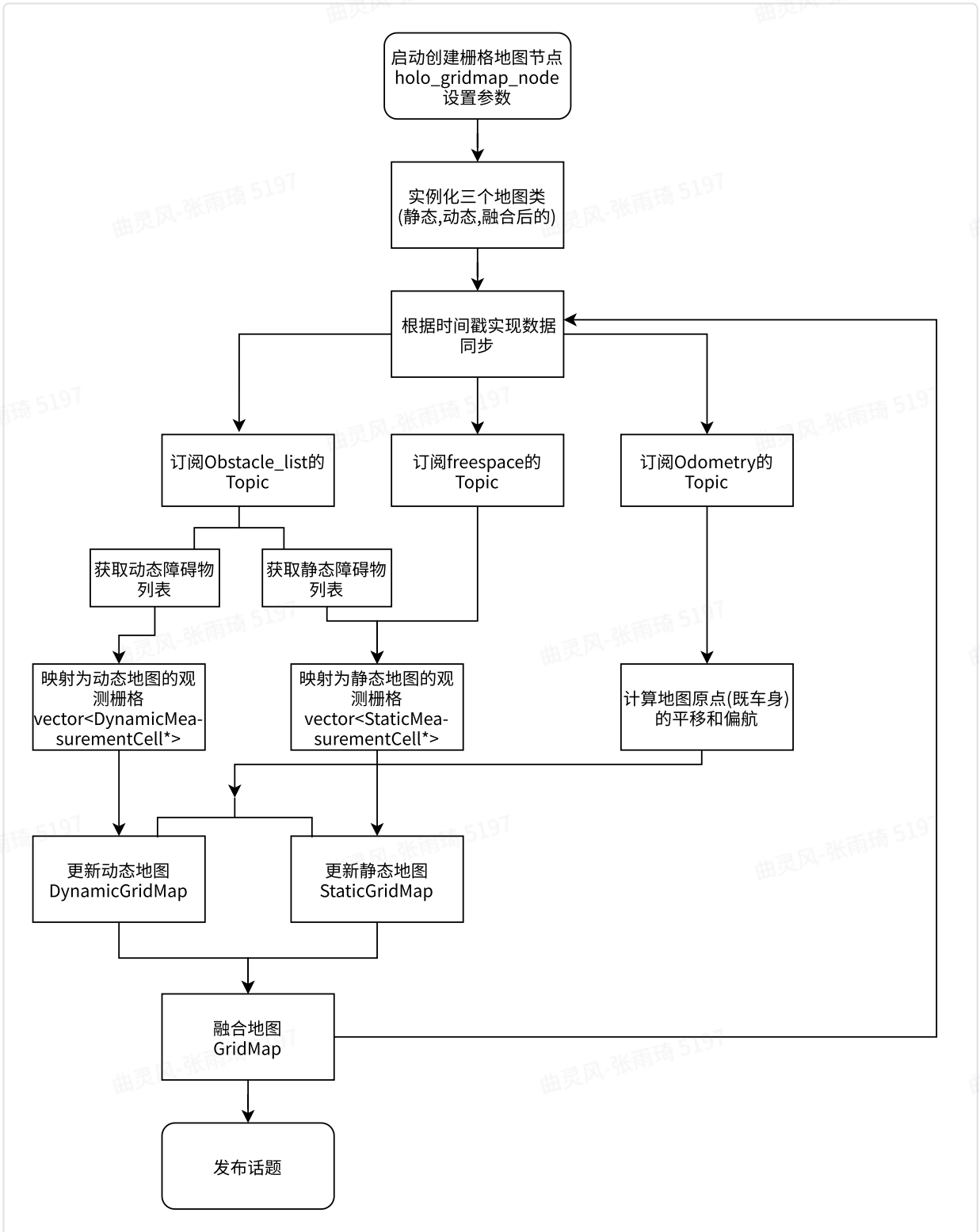
3.1 静态地图栅格

```
struct StaticGridMap
{
  float occ_ratio_; //占用情况
  Classification classification_; //类别
};
Struct StaticMeasurementCell
{
    float occ_ratio_; //占用情况
    Classification classification_; 类别
    Point2d_grid point_;  //在地图中的离散位置
};
```

3.2 动态地图栅格(待更新)

```
struct DynamicGridMap
{Struct
};
Struct DynamicMeasurementCell
{
};
```

二.具体流程



三.具体工作

基础任务	1.创建接收和发送节点		2.根据时间戳同步数据		3.根据obstacle_list区分动态静态障碍物	
映射	1.freespace		2.bounding		2.根据Odometry计算车	

	映射为栅格		box映射为栅格		辆平移和旋转矩阵	
观测栅格列表	2.定义类和功能		1.生成静态观测栅格列表		2.生成动态观测栅格列表	
静态栅格地图	1.定义类和功能		2.初始化		3.更新(包括根据观测值更新占用状态和根据平移和和旋转矩阵更新栅格下标)	
动态栅格地图	1.定义类和功能		2.初始化栅格		3.初始化粒子	
	3.更新粒子(持续的,新生的,死亡的)		4.粒子重采样		5.更新栅格占用状态	
融合动静态地图	1.定义类和功能(和下游规划协定好)		2.动静态地图融合		3.发布地图(转为msg)	