

```

gtsam::PriorFactor<gtsam::Pose3> priorPose(X(0),
prevPose_, updatedPoseNoise);
graphFactors.add(priorPose);
// 把上一次优化出的速度作为重新初始化的priorVel
gtsam::PriorFactor<gtsam::Vector3> priorVel(V(0),
prevVel_, updatedVelNoise);
graphFactors.add(priorVel);
// 把上一次优化出的bias作为重新初始化的priorBias
gtsam::PriorFactor<gtsam::imuBias::ConstantBias>
priorBias(B(0), prevBias_, updatedBiasNoise);
graphFactors.add(priorBias);
// 将prior状态设置成初始估计值
graphValues.insert(X(0), prevPose_);
graphValues.insert(V(0), prevVel_);
graphValues.insert(B(0), prevBias_);
这个graphvalue和graphnode什么区别，

```

下面一句话先给你一个最核心的理解：

在 GTSAM 里：Factor Graph = Nodes（变量） + Factors（约束）；
Values 是“变量的当前取值（初始解或优化解）”，而不是图本身的一部分。

因此：


- **graphFactors（或 graph）** —— 存的是 **Factor（约束）**
- **graphNodes（变量）** —— 实际上在 GTSAM 不直接存在一个叫“graphNodes”的结构，节点由 **Keys** 决定（例如 X(0), V(0), B(0) 这些 key）。
- **graphValues（Values）** —— 存的是 **变量的值**（初始估计值或者优化后的结果）

你代码里的三个东西分别是什么？

1. graphFactors（又叫 NonlinearFactorGraph）

它只存“因子”（约束），例如：

cpp

 复制代码

```
gtsam::PriorFactor<gtsam::Pose3> priorPose(...)  
graphFactors.add(priorPose);
```


因子使用 key（如 X(0), V(0)）来连接图中的变量，但它不管变量值是多少。

2. graphNodes（概念，不是实际结构）

GTSAM 的图节点其实由 factor 的 key 决定：

例如：

cpp

 复制代码

```
priorPose connects to X(0)  
priorVel connects to V(0)  
priorBias connects to B(0)
```


你一旦给一个 key 添加 factor，它就算是图里的一个节点。

GTSAM 里没有一个独立的 Node 容器。

3. graphValues（Values）

这是一个 **Key → Value** 的字典，存每个变量的“当前估计”：

cpp

 复制代码

```
graphValues.insert(X(0), prevPose_);  
graphValues.insert(V(0), prevVel_);  
graphValues.insert(B(0), prevBias_);
```

为什么必须提供 Values？

因为优化器需要一个“初始猜测”才能开始求解。

总结成一句话

名称	在 GTSAM 的作用
FactorGraph / graphFactors	存因子（约束），定义变量之间的关系
Node（隐含概念）	由 key 决定的变量，不需要单独创建
Values / graphValues	变量的具体取值（初值或优化结果）

 **更形象的理解（非常重要）**

把 GTSAM 想成一个“解方程”的框架：

- **Factor** = 方程约束
- **Key** = 方程中的未知量名字
- **Values** = 未知量的当前猜测

优化器在“方程（factor）”和“变量的初猜（values）”之间求出最优解。

如果你需要，我还能帮你画一个 ASCII 图讲清楚 factor、node、values 的结构关系，会更形象。

     ...

先验观测约束后，原有node也要有一个初始估计值吗

要的！必须要有！

即使你给某个变量（node）加了先验因子（prior factor），它仍然需要在 **Values** 里提供一个初始估计值，否则优化器无法启动。

 **为什么“加了先验因子”还要初始估计？**

因为：

- 先验因子（**PriorFactor**）只是一个约束（告诉优化器：这个变量应该接近某个值）

- **Values 才是未知量的初始猜测**

GTSAM 是一个非线性优化器，它必须从某个初始值开始迭代。

所以：


加因子 != 给初始值

两者作用完全不同。

✓ 举个例子（你代码中的情况）

你加了先验因子：


cpp

 复制代码

```
gtsam::PriorFactor<gtsam::Pose3> priorPose(X(0), prevPose_, noise);  
graphFactors.add(priorPose);
```

但你还必须：


cpp

 复制代码

```
graphValues.insert(X(0), prevPose_);
```

否则 optimizer 会报以下错误：

pgsql

 复制代码

```
Values has no initial estimate for key X(0)
```

那如果我 不给 *prior factor*，只给初始值行不行？

可以，但这时变量只是 有一个起点，没有被约束。

优化器会根据其他因子（例如 IMU、GPS、Lidar）来推它。

那如果我 给了 *prior factor*, 但不给初始值会怎样?

会报错。

因为 *prior factor* 只定义“约束”，不定义“初始估计”。

优化器需要在迭代的起点就知道：“变量现在的值是多少，用来开始求解”。

更形象地理解

可以把 *factor graph* 看成下面这样：

- **Factor**：方程
- **Values**：方程里未知数的初始值
- **Optimizer**：拿着初值，不断更新变量

如果你只有方程没有初值，优化器无从下手。

✓ 结论（最重要一句）

任何出现在因子图中的变量（**node**）都必须在 **Values** 里提供初始估计，无论它有没有 **prior factor**。

prior 只是一个约束，不是初始估计。

如果你愿意，我可以再画一个更直观的图，明确对比：

- *factor*
- *node(key)*
- *values*

 之间的关系。  ...

你喜欢此风格吗？

