1. **ORB 特征点**

   参考 workspace-submit/01-ORB/computeORB.cpp

   1. 为什么说 ORB 是一种二进制特征?

   因为 ORB 特征描述均由 0 与 1 组成，没有实数。

   2. 为什么在匹配时使用 50 作为阈值,取更大或更小值会怎么样?

   该数值来自高博的工程经验。增大阈值，匹配对数会增加；减小阈值，匹配对数会减少。

   3. 暴力匹配在你的机器上表现如何?你能想到什么减少计算量的匹配方法吗?

   在我的 Alienware 上运行时间依然非常长。为减少运算量，可使用 Locality Sensitive Hashing(LSH)或者 Fast Library for Approximated Nearest Neighbor(FLANN)

2. **从 E 恢复 R, t**

   参考 workspace-submit/02-camera-pose-estimation/pose-estimation.cpp

3. 用 G-N 实现 Bundle Adjustment 中的位姿估计

参考 workspace-submit/03-pnp-using-bundle-adjustment/GN-BA.cpp

1. 如何定义重投影误差?

```
// rigid transform defined by SE3:
auto R = T_esti.rotation_matrix();
auto t = T_esti.translation();
// point in camera frame:
auto p_camera = R * p3d[i] + t;

// point in pixel frame:
auto p_pixel = K * p_camera;

// error:
auto e = p2d[i];
e.x() -= p_pixel.x() / p_pixel.z();
e.y() -= p_pixel.y() / p_pixel.z();
```

2. 该误差关于自变量的雅可比矩阵是什么?

```
double X_prime = p_camera.x();
double Y_prime = p_camera.y();
double Z_prime = p_camera.z();
Matrix<double, 2, 6> J;
J << \
                                        fx/Z_prime,                                                0.0, \
                    -fx*X_prime/(Z_prime*Z_prime),         -fx*X_prime*Y_prime/(Z_prime*Z_prime), \
    fx*(1 + (X_prime*X_prime)/(Z_prime*Z_prime)),                          -fx*Y_prime/Z_prime, \
                                                0.0,                                       fy/Z_prime, \
                    -fy*Y_prime/(Z_prime*Z_prime), -fy*(1 + (Y_prime*Y_prime)/(Z_prime*Z_prime)), \
            fy*(X_prime*Y_prime)/(Z_prime*Z_prime),                               fy*X_prime/Z_prime
;
J *= -1.0;
```

3. 解出更新量之后,如何更新至之前的估计上?

```
// update your estimation
T_esti = Sophus::SE3::exp(dx) * T_esti;
```

4. 用 ICP 实现轨迹对齐

参考 workspace-submit/04-ICP/trajectory-matching.cpp