深蓝学院
shenlanxueyuan.com

视觉SLAM理论与实践

第七次作业讲评

主讲人 田智豪

2 Bundle Adjustment

1. 为何说 Bundle Adjustment is slow 是不对的?
答: 因为利用 H 矩阵的稀疏结构可以加速优化

2. BA 中有哪些需要注意参数化的地方?Pose 和 Point 各有哪些参数化方式?有何优缺点。
答:需要参数化的地方: 相机位姿, 相机内参, 三维点 p和其投影后的像素坐标

pose: 三维欧拉角+三维位移,四元数+三维位移,旋转矩阵+三维位移,变换矩阵等. 欧拉角-万向锁问题。旋转矩阵-直观, 但自由度过多, 小旋转给不出确切矩阵。四元数-参数简洁, 不直观。

point: homogeneous affine(X,Y,Z,1) - 直观, 但是需要参数有较大的改变才能明显的影响到 cost function。homogeneous projective parametrization(X,Y,Z,W) - 可以表示距离无穷远的点

2 Bundle Adjustment

3. * 本文写于 2000 年,但是文中提到的很多内容在后面十几年的研究中得到了印证。你能看到哪些方向在后续工作中有所体现?请举例说明。

答: 3.4 节的Intensity-based methods 就是 BA 在直接法中的应用。
第 5 节 Network Structure 可以对应到 SLAM 中的图优化模型。
H 的稀疏性可以实现 BA 实时，在 07 年的 PTAM 上实现。

2 Bundle Adjustment

深蓝学院
shenlanxueyuan.com

```cpp
typedef g2o::BlockSolver<g2o::BlockSolverTraits<9, 3>> BalBlockSolver;
typedef Eigen::Matrix<double, 9, 1> Vector9d;

int main(int argc, char **argv)
{
    std::string filename1 = "../problem-16-22106-pre.txt";
    std::string filename2 = "../inital.ply";
    std::string filename3 = "../final.ply";

    BALProblem bal_problem(filename1);

    g2o::SparseOptimizer optimizer;

    g2o::LinearSolver<BalBlockSolver::PoseMatrixType>* linearSolver =
            new g2o::LinearSolverCSparse<BalBlockSolver::PoseMatrixType>();
    dynamic_cast<g2o::LinearSolverCSparse<BalBlockSolver::PoseMatrixType>* >(linearSolver)->setBlockOrdering(true);

    //矩阵块求解器
    BalBlockSolver* solver_ptr = new BalBlockSolver(std::unique_ptr<BalBlockSolver::LinearSolverType>(linearSolver));

    //使用LM算法
    g2o::OptimizationAlgorithmLevenberg* solver = new g2o::OptimizationAlgorithmLevenberg(std::unique_ptr<BalBlockSolver>(solver_ptr));
    //优化器构建完成
    optimizer.setAlgorithm(solver);

    buildGraph(&bal_problem, &optimizer);

    optimizer.initializeOptimization();
    optimizer.setVerbose(true);
    optimizer.optimize(20);

    saveResult(&bal_problem, &optimizer, filename3);
}
```

程序基本流程

2 Bundle Adjustment

```
class BALProblem
{
public:
    BALProblem(const std::string &filename);
    ~BALProblem()
    {
        delete[] point_index;
        delete[] camera_index;
        delete[] observations;
        delete[] c_param;
        delete[] p_param;
    }

    int num_cameras;
    int num_points;
    int num_observations;

    int *point_index;
    int *camera_index;
    double *observations;
    //读取时候有调换， c_param是先平移再旋转。
    double *c_param;
    double *p_param;
    void WriteToPLYFile(const std::string &filename) const;
};

template <typename T>
void FscanfOrDie(FILE *fptr, const char *format, T *value)
{
    int num_scanned = fscanf(fptr, format, value);
    if (num_scanned != 1)
        std::cerr << "Invalid UW data file. ";
}
```

BALProblem类

2 Bundle Adjustment

深蓝学院
shenlanxueyuan.com

```
BALProblem::BALProblem(const std::string &filename)
{
    //构造函数的任务：存储文件的全部信息
    //检查文件是否正确打开
    FILE *fptr = fopen(filename.c_str(), "r");
    if (fptr == NULL)
    {
        std::cerr << "Error: unable to open file " << filename;
        return;
    };
    //读取第一行的数据，num_cameras 个相机，num_points 个路标，num_observations 个观测
    FscanfOrDie(fptr, "%d", &num_cameras);
    FscanfOrDie(fptr, "%d", &num_points);
    FscanfOrDie(fptr, "%d", &num_observations);

    std::cout << "Header: " << num_cameras
              << " " << num_points
              << " " << num_observations << std::endl;
    //每行四个数据，相机index，路标index，观测到的2D坐标
    point_index = new int[num_observations];
    camera_index = new int[num_observations];
    observations = new double[2 * num_observations];

    c_param = new double[9 * num_cameras];
    p_param = new double[3 * num_points];

    //index下标从0开始，先保存observation
    for (int i = 0; i < num_observations; ++i)
    {
        FscanfOrDie(fptr, "%d", camera_index + i);
        FscanfOrDie(fptr, "%d", point_index + i);
        FscanfOrDie(fptr, "%lf", observations + 2 * i);
        FscanfOrDie(fptr, "%lf", observations + 2 * i + 1);
    }
    //再保存相机和3D点的参数，作为初始值
    for (int i = 0; i < 9 * num_cameras; ++i)
    {
        FscanfOrDie(fptr, "%lf", c_param + i);
    }

    for (int i = 0; i < 3 * num_points; ++i)
    {
        FscanfOrDie(fptr, "%lf", p_param + i);
    }

    fclose(fptr);
}
```

2 Bundle Adjustment

```cpp
void BALProblem::WriteToPLYFile(const std::string &filename) const
{
    std::ofstream of(filename.c_str());
    //PLY文件的基本格式
    of << "ply"
       << '\n'
       << "format ascii 1.0"
       << '\n'
       << "element vertex " << num_cameras + num_points
       << '\n'
       << "property float x"
       << '\n'
       << "property float y"
       << '\n'
       << "property float z"
       << '\n'
       << "property uchar red"
       << '\n'
       << "property uchar green"
       << '\n'
       << "property uchar blue"
       << '\n'
       << "end_header" << std::endl;

    //首先写入相机坐标 以及 SE3 到 普通坐标系的转换
    for (int i = 0; i < num_cameras; i++)
    {
        //没有必要调换顺序吧？ 本程序默认前三个参数是平移，后三个是旋转。符合Sophus的顺序。
        Eigen::Map<Eigen::Matrix<double, 9, 1>> param(c_param + i * 9, 9);
        //提取外参
        Sophus::Vector6d se3;
        se3 << param(3), param(4), param(5),param(0), param(1), param(2);
        //外参的逆就是相机在世界坐标系的坐标
        Eigen::Matrix4d Tcw = Sophus::SE3::exp(se3).matrix().inverse();

        of << Tcw(0, 3) << ' ' << Tcw(1, 3) << ' ' << Tcw(2, 3)
           << " 0 255 0" << '\n';
    }

    for (int i = 0; i < num_points; i++)
    {
        Eigen::Map<Eigen::Vector3d> param(p_param + i * 3, 3);
        of << param(0) << ' ' << param(1) << ' ' << param(2)
           << " 255 255 255" << '\n';
    }
}
```

2 Bundle Adjustment


```
void buildGraph(BALProblem *bal_problem, g2o::SparseOptimizer *optimizer)
{
    // 构造图模型 开始 >>>>>
    const int num_points = bal_problem->num_points;
    const int num_cameras = bal_problem->num_cameras;
    const int num_observations = bal_problem->num_observations;
    const double *observations = bal_problem->observations;
    const double *c_param = bal_problem->c_param;
    const double *p_param = bal_problem->p_param;

    //添加相机顶点
    for (int i = 0; i < num_cameras; ++i)…

    //添加3D点的顶点
    for (int i = 0; i < num_points; ++i)
    {
        //param保存了相机和3D点的参数， 所以起始位置要调整
        Eigen::Vector3d temVecPoint;
        for (int j = 0; j < 3; j++)
        {
            temVecPoint(j) = p_param[3 * i + j];
        }

        VertexPointBAL *pPoint = new VertexPointBAL();
        pPoint->setEstimate(temVecPoint);
        pPoint->setId(i + num_cameras);
        //without this line -> error at block_solver.hpp  134 , can't build structure
        pPoint->setMarginalized(true);
        optimizer->addVertex(pPoint);
    }

    //添加边
    for (int i = 0; i < num_observations; ++i)…
}
```

构造优化图。往optimizer里面添加顶点和边。

2 Bundle Adjustment

深蓝学院
shenlanxueyuan.com

```
//定义3D点的vertex
class VertexPointBAL : public g2o::BaseVertex<3, Eigen::Vector3d>
{
public:
    EIGEN_MAKE_ALIGNED_OPERATOR_NEW;

    VertexPointBAL() {}

    virtual bool read(std::istream & /*is*/) { return false; }
    virtual bool write(std::ostream & /*os*/) const { return false; }

    virtual void setToOriginImpl() {}

    virtual void oplusImpl(const double *update)
    {
        Eigen::Vector3d::ConstMapType x(update);
        _estimate += x;
    }
};
```

自定义的point顶点

2 Bundle Adjustment

深蓝学院 shenlanxueyuan.com

```
class EdgeObservationBAL : public g2o::BaseBinaryEdge<2, Eigen::Vector2d, VertexCameraBAL, VertexPointBAL>
{
public:
    EIGEN_MAKE_ALIGNED_OPERATOR_NEW;
    EdgeObservationBAL(){};

    virtual bool read(std::istream & /*is*/) { return false; }

    virtual bool write(std::ostream & /*os*/) const { return false; }

    virtual void computeError() override…


    virtual void linearizeOplus() override…
};
```

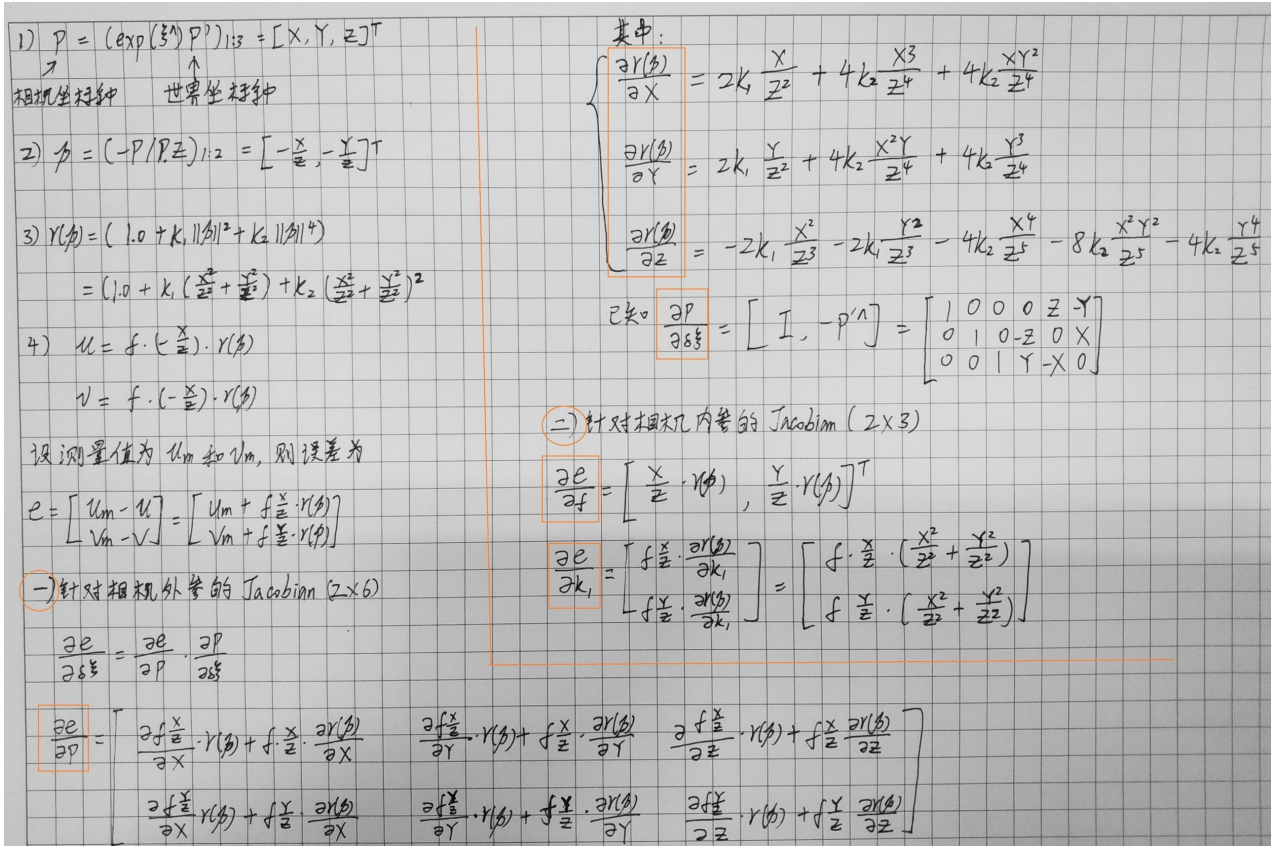
```
//定义相机参数的vertex
class VertexCameraBAL : public g2o::BaseVertex<9, Eigen::Matrix<double, 9, 1>>
{
public:
    //不添加这一句就会出现错误
    //main.cpp:(.text+0x443)：对‘VertexCameraBAL::VertexCameraBAL()’未定义的引用
    EIGEN_MAKE_ALIGNED_OPERATOR_NEW;

    VertexCameraBAL() {}

    virtual bool read(std::istream & /*is*/) { return false; }
    virtual bool write(std::ostream & /*os*/) const { return false; }

    //设定成0试试？
    virtual void setToOriginImpl() {}

    virtual void oplusImpl(const double *update)…
};
```

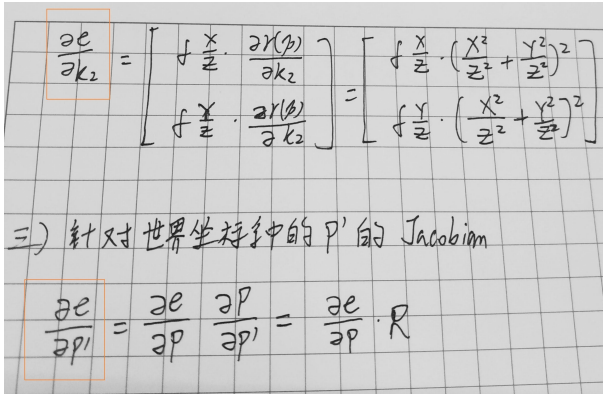
自定义相机顶点和连接 相机与空间点 的边

2 Bundle Adjustment

深蓝学院 shenlanxueyuan.com

雅克比的数学推导





3 直接法的 Bundle Adjustment

3.1 数学模型

1. 如何描述任意一点投影在任意一图像中形成的 error?

   答:$error = I(p) - I(\pi(KTp))$

2. 每个 error 关联几个优化变量?

   答:每个误差项 error 关联两个优化变量,分别为:P_w, ξ

3. error 关于各变量的雅可比是什么?

$$\boldsymbol{J} = -\frac{\partial \boldsymbol{I}_2}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \delta \boldsymbol{\xi}}. \qquad \frac{\partial \boldsymbol{u}}{\partial \delta \boldsymbol{\xi}} = \begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{f_x X}{Z^2} & -\frac{f_x XY}{Z^2} & f_x + \frac{f_x X^2}{Z^2} & -\frac{f_x Y}{Z} \\ 0 & \frac{f_y}{Z} & -\frac{f_y Y}{Z^2} & -f_y - \frac{f_y Y^2}{Z^2} & \frac{f_y XY}{Z^2} & \frac{f_y X}{Z} \end{bmatrix}$$

3 直接法的 Bundle Adjustment

3.2 实现

1. 能否不要以 [x, y, z] T 的形式参数化每个点?
答: 能,如题目中说 的,可以采用逆深度参数化 。

2. 取 4x4 的 patch 好吗?取更大的 patch 好还是取小一点的 patch 好?
答: 从结果看 4*4 可以。固定场景的话可能更大一点好, 但会增加运算量。

3. 从本题中,你看到直接法与特征点法在 BA 阶段有何不同?
答: 误差计算方法不同。重投影误差和 灰度误差.

4. 由于图像的差异,你可能需要鲁棒核函数,例如 Huber。此时 Huber 的阈值如何选取?
答: 假设误差项是高斯分布的 , 则误差项的平方服从卡方分布 , 然后确定误差项的自由度 , 以及置信度 , 根据自由度
和置信度查找卡方分布表就能知道阈值是多少 , 一般置信度假设 0.95

3 直接法的 Bundle Adjustment


```cpp
virtual void computeError() override
{
    // TODO START YOUR CODE HERE
    // compute projection error ...
    const g2o::VertexSBAPointXYZ *vertexPw = static_cast<const g2o::VertexSBAPointXYZ *>(vertex(0));
    const VertexSophus *vertexTcw = static_cast<const VertexSophus *>(vertex(1));
    const Eigen::Vector3d point = vertexPw->estimate();
    const Sophus::SE3 T = vertexTcw->estimate();
    Eigen::Vector3d Pc = T * point;

    float u = fx * Pc(0) / Pc(2) + cx;
    float v = fy * Pc(1) / Pc(2) + cy;

    int cols = targetImg.cols;
    int rows = targetImg.rows;
    if (u - 2 < 0 || u + 1 > cols - 1 || v - 2 < 0 || v + 1 > rows - 1)
    {
        for (int i = 0; i < 16; i++)
        {
            _error[i] = 0;
        }
    }
    else
    {
        for (int x = -2; x <= 1; x += 1)
        {
            for (int y = -2; y <= 1; y += 1)
            {
                int num = 4 * x + y + 10;
                _error[num] = origColor[num] - GetPixelValue(targetImg, u + x, v + y);
            }
        }
    }
    // END YOUR CODE HERE
}
```

3 直接法的 Bundle Adjustment

深蓝学院 shenlanxueyuan.com

```
// TODO add vertices, edges into the graph optimizer
// START YOUR CODE HERE
for (int i = 0; i < poses.size(); i++)
{
    VertexSophus *v = new VertexSophus();
    v->setId(i);
    v->setEstimate(poses[i]);
    optimizer.addVertex(v);
}
for (int i = 0; i < points.size(); i++)
{
    g2o::VertexSBAPointXYZ *v = new g2o::VertexSBAPointXYZ();
    v->setId(i + poses.size());
    v->setEstimate(points[i]);
    v->setMarginalized(true);
    optimizer.addVertex(v);
}

for (int i = 0; i < poses.size(); i++)
{
    for (int j = 0; j < points.size(); j++)
    {
        EdgeDirectProjection *edge = new EdgeDirectProjection(color[j], images[i]);
        edge->setVertex(1, dynamic_cast<VertexSophus *>(optimizer.vertex(i)));
        edge->setVertex(0, dynamic_cast<g2o::VertexSBAPointXYZ *>(optimizer.vertex(j + poses.size())));
        edge->setInformation(Matrix16d::Identity());
        edge->setRobustKernel(new g2o::RobustKernelHuber());
        optimizer.addEdge(edge);
    }
}
// END YOUR CODE HERE
```

2 Bundle Adjustment


```
// TODO fetch data from the optimizer
// START YOUR CODE HERE
for (int i = 0; i < poses.size(); i++)
{
    VertexSophus *v = dynamic_cast<VertexSophus *>(optimizer.vertex(i));
    poses[i] = v->estimate();
}

for (int i = 0; i < points.size(); i++)
{
    g2o::VertexSBAPointXYZ *v = dynamic_cast<g2o::VertexSBAPointXYZ *>(optimizer.vertex(i + poses.size()));
    points[i] = v->estimate();
}
// END YOUR CODE HERE
```

深蓝学院
shenlanxueyuan.com

感谢各位聆听

Thanks for Listening