2018 年第 4 次课习题答案

——Vance 吴方熠

2 图像去畸变（3 分，约 1 小时）

现实生活中的图像总存在畸变。原则上来说，针孔透视相机应该将三维世界中的直线投影成直线，但是当我们使用广角和鱼眼镜头时，由于畸变的原因，直线在图像里看起来是扭曲的。本次作业，你将尝试如何对一张图像去畸变，得到畸变前的图像。

图 1: 测试图像

图 1 是本次习题的测试图像（code/test.png），来自 EuRoC 数据集 [1]。可以明显看到实际的柱子、箱子的直线边缘在图像中被扭曲成了曲线。这就是由相机畸变造成的。根据我们在课上的介绍，畸变前后的坐标变换为：

$$\begin{cases} x_{\text{distorted}} = x\left(1 + k_1 r^2 + k_2 r^4\right) + 2p_1 xy + p_2\left(r^2 + 2x^2\right) \\ y_{\text{distorted}} = y\left(1 + k_1 r^2 + k_2 r^4\right) + p_1\left(r^2 + 2y^2\right) + 2p_2 xy \end{cases} \tag{1}$$

其中 x, y 为去畸变后的坐标，$x_{\text{distorted}}, y_{\text{distroted}}$ 为去畸变前的坐标。现给定参数：

$$k_1 = -0.28340811,\ k_2 = 0.07395907,\ p_1 = 0.00019359,\ p_2 = 1.76187114e-05.$$

以及相机内参

$$f_x = 458.654,\ f_y = 457.296,\ c_x = 367.215,\ c_y = 248.375.$$

请根据 undistort_image.cpp 文件中内容，完成对该图像的去畸变操作。

答：

图像去畸变过程：

1) 用书中（5.14）式算出无畸变图各像素点对应的理论坐标 x_ud, y_ud;
2) 计算坐标点到原点的距离 r2 = x_ud*x_ud + y_ud*y_ud;
3) 根据参数，由（5.13）式算出各点的畸变坐标；
4) 再由（5.14）式算出各坐标对应的畸变像素坐标（即在畸变图上的位置，不一定为整数）
5) 校正图像，校正后的图像各像素点的灰度值等于该像素点对应的畸变像素坐标的灰度值。（注意可用最近邻差值法解决像素坐标不为整数的问题）

自编代码部分如下所示：

```
    // start your code here
    double x_ud = (u - cx)/fx;
    double y_ud = (v - cy)/fy;
    double r2 = x_ud*x_ud + y_ud*y_ud;
    double r4 = r2*r2;
    double x_d = x_ud*(1 + k1*r2 + k2*r4) + 2*p1*x_ud*y_ud + p2*(r2 + 2*x_ud*x_ud);
    double y_d = y_ud*(1 + k1*r2 + k2*r4) + p1*(r2 + 2*y_ud*y_ud) + 2*p2*x_ud*y_ud;

    u_distorted = fx*x_d + cx;
    v_distorted = fy*y_d + cy;
    // end your code here
```

程序运行的结果截图如下所示：

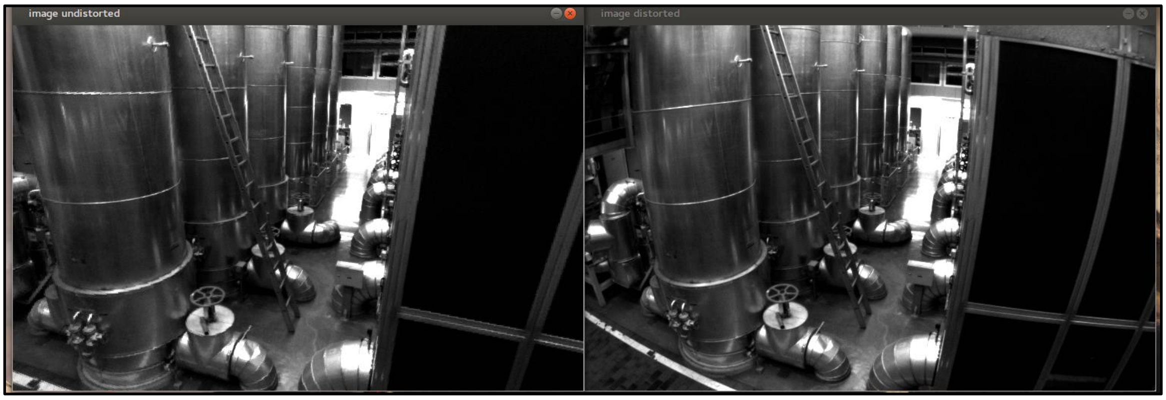

图 1 图像去畸变（左）和未去畸变（右）的效果

3 双目视差的使用 （2 分，约 1 小时）

双目相机的一大好处是可以通过左右目的视差来恢复深度。课程中我们介绍了由视差计算深度的过程。本题，你需要根据视差计算深度，进而生成点云数据。本题的数据来自 Kitti 数据集 [2]。

Kitti 中的相机部分使用了一个双目模型。双目采集到左图和右图，然后我们可以通过左右视图恢复出深度。经典双目恢复深度的算法有 BM(Block Matching), SGBM(Semi-Global Matching)[3, 4] 等，但本题不探讨立体视觉内容（那是一个大问题）。我们假设双目计算的视差已经给定，请你根据双目模型，画出图像对应的点云，并显示到 Pangolin 中。

本题给定的左右图见 code/left.png 和 code/right.png，视差图亦给定，见 code/right.png。双目的参数如下：

$$f_x = 718.856,\ f_y = 718.856,\ c_x = 607.1928,\ c_y = 185.2157.$$

且双目左右间距（即基线）为：

$$d = 0.573 \text{ m}.$$

请根据以上参数，计算相机数据对应的点云，并显示到 Pangolin 中。程序请参考 code/disparity.cpp 文件。

答：

自编代码部分如下所示：

```
......
    // 间距 baseline
    double b = 0.573;
......
            // start your code here (~6 lines)
            // 根据双目模型计算 point 的位置
            point[2] = fx * b / disparity.at<uchar>(v, u);
            point[0] = (u - cx) * point[2] / fx;
            point[1] = (v - cy) * point[2] / fy;

            pointcloud.push_back(point);
            // end your code here
        }
......
```

程序运行结果如下所示：

图 2 由双目视差计算出的深度图效果

4 矩阵运算微分（2 分，约 1.5 小时）

在优化中经常会遇到矩阵微分的问题。例如，当自变量为向量 $\mathbf{x}$，求标量函数 $u(\mathbf{x})$ 对 $\mathbf{x}$ 的导数时，即为矩阵微分。通常线性代数教材不会深入探讨此事，这往往是矩阵论的内容。我在 ppt/目录下为你准备了一份清华研究生课的矩阵论课件（仅矩阵微分部分）。阅读此 ppt，回答下列问题：

设变量为 $\mathbf{x} \in \mathbb{R}^N$，那么：

1. 矩阵 $\mathbf{A} \in \mathbb{R}^{N\times N}$，那么 $\mathrm{d}(\mathbf{Ax})/\mathrm{d}\mathbf{x}$ 是什么[1]?
2. 矩阵 $\mathbf{A} \in \mathbb{R}^{N\times N}$，那么 $\mathrm{d}(\mathbf{x}^{\mathrm{T}}\mathbf{Ax})/\mathrm{d}\mathbf{x}$ 是什么?
3. 证明：

$$\mathbf{x}^{\mathrm{T}}\mathbf{Ax} = \mathrm{tr}(\mathbf{Axx}^{\mathrm{T}}). \tag{2}$$

解：

1. 常数矩阵 A 与向量的微分

$$\frac{\mathrm{d}(\boldsymbol{A}\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \frac{\boldsymbol{A}^{\mathrm{T}}\mathrm{d}\boldsymbol{x}^{\mathrm{T}}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \boldsymbol{A}^{\mathrm{T}}$$

2. 二次型运算

$$\frac{\mathrm{d}(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x})}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \frac{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\mathrm{d}\boldsymbol{x}}{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}} = \boldsymbol{A}\boldsymbol{x} + (\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A})^{\mathrm{T}} = (\boldsymbol{A} + \boldsymbol{A}^{\mathrm{T}})\boldsymbol{x}$$

3. 证明：

将两侧同时对常数矩阵 A 取导，有：

$$左边 = \frac{\mathrm{d}(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x})}{\mathrm{d}\boldsymbol{A}} = (\boldsymbol{x}^{\mathrm{T}})^{\mathrm{T}}\boldsymbol{x}^{\mathrm{T}} = \boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}$$

$$右边 = \frac{\mathrm{d}\left(\mathbf{tr}(\boldsymbol{A}\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}})\right)}{\mathrm{d}\boldsymbol{A}} = (\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}})^{\mathrm{T}} = \boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}$$

因为求导的 A 为常数矩阵，故有

$$\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} = \mathbf{tr}(\boldsymbol{A}\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}})$$

得证。

5 高斯牛顿法的曲线拟合实验（3 分，约 2 小时）

我们在课上演示了用 Ceres 和 g2o 进行曲线拟合的实验，可以看到优化框架给我们带来了诸多便利。本题中你需要自己实现一遍高斯牛顿的迭代过程，求解曲线的参数。我们将原题复述如下。设有曲线满足以下方程：

$$y = \exp(ax^2 + bx + c) + w. \tag{3}$$

其中 a, b, c 为曲线参数，w 为噪声。现有 N 个数据点 (x, y)，希望通过此 N 个点来拟合 a, b, c。实验中取 $N = 100$。

那么，定义误差为 $e_i = y_i - \exp(ax_i^2 + bx_i + c)$，于是 (a, b, c) 的最优解可通过解以下最小二乘获得：

$$\min_{a,b,c} \frac{1}{2} \sum_{i=1}^{N} \left\| y_i - \exp\left(ax_i^2 + bx_i + c\right) \right\|^2. \tag{4}$$

现在请你书写 Gauss-Newton 的程序以解决此问题。程序框架见 code/gaussnewton.cpp，请填写程序内容以完成作业。作为验证，按照此程序的设定，估计得到的 a, b, c 应为：

$$a = 0.890912,\ b = 2.1719,\ c = 0.943629.$$

这和书中的结果是吻合的。

答：

自编代码部分如下所示：

```
......
        for (int i = 0; i < N; i++) {
            double xi = x_data[i], yi = y_data[i];  // 第 i 个数据点
            // start your code here
            double error = 0;   // 第 i 个数据点的计算误差
            error = yi - exp(ae*xi*xi + be*xi + ce);    // 填写计算 error 的表达式
            Vector3d J;         // 雅可比矩阵
            J[0] = -exp(ae*xi*xi + be*xi + ce)*xi*xi;   // de/da
            J[1] = -exp(ae*xi*xi + be*xi + ce)*xi;      // de/db
            J[2] = -exp(ae*xi*xi + be*xi + ce);         // de/dc

            H += J * J.transpose(); // GN 近似的 H
            b += -error * J;
            // end your code here

            cost += error * error;
        }

        // 求解线性方程 Hx=b，建议用 ldlt
        // start your code here
        Vector3d dx;
        dx = H.ldlt().solve(b);
        // end your code here
......
```

程序运行结果如下所示：

```
vance@Fai-PC: ~/slam_ws/slambook/class/04/L4/code/build-qt
File  Edit  View  Search  Terminal  Help
vance@Fai-PC:~/slam_ws/slambook/class/04/L4/code/build-qt$ ./gaussnewton
total cost: 3.19575e+06
total cost: 376785
total cost: 35673.6
total cost: 2195.01
total cost: 174.853
total cost: 102.78
total cost: 101.937
total cost: 101.937
total cost: 101.937
total cost: 101.937
total cost: 101.937
cost: 101.937, last cost: 101.937
estimated abc = 0.890912, 2.1719, 0.943629
vance@Fai-PC:~/slam_ws/slambook/class/04/L4/code/build-qt$
```

结果和答案吻合。

6 * 批量最大似然估计（2 分，约 2 小时）

考虑离散时间系统：

$$
\begin{aligned}
x_k &= x_{k-1} + v_k + w_k, & w &\sim \mathcal{N}(0, Q) \\
y_k &= x_k + n_k, & n_k &\sim \mathcal{N}(0, R)
\end{aligned}
$$

这可以表达一辆沿 x 轴前进或后退的汽车。第一个公式为运动方程，v_k 为输入，w_k 为噪声；第二个公式为观测方程，y_k 为路标点。取时间 $k = 0, \ldots, 3$，现希望根据已有的 v, y 进行状态估计。设初始状态 x_0 已知。

请根据本题题设，推导批量（batch）最大似然估计。首先，令批量状态变量为 $\mathbf{x} = [x_0, x_1, x_2, x_3]^{\mathrm{T}}$，令批量观测为 $\mathbf{z} = [x_0, v_1, v_2, v_3, y_1, y_2, y_3]^{\mathrm{T}}$，那么：

1. 可以定义矩阵 $\mathbf{H}$，使得批量误差为 $\mathbf{e} = \mathbf{z} - \mathbf{H}\mathbf{x}$。请给出此处 $\mathbf{H}$ 的具体形式。
2. 据上问，最大似然估计可转换为最小二乘问题：

$$
\mathbf{x}^* = \arg\min \frac{1}{2} (\mathbf{z} - \mathbf{H}\mathbf{x})^{\mathrm{T}} \mathbf{W}^{-1} (\mathbf{z} - \mathbf{H}\mathbf{x}), \tag{5}
$$

   其中 $\mathbf{W}$ 为此问题的信息矩阵，可以从最大似然的概率定义给出。请给出此问题下 $\mathbf{W}$ 的具体取值。
3. 假设所有噪声相互无关，该问题存在唯一的解吗？若有，唯一解是什么？若没有，说明理由。

解：

参考《State Estimation for Robotics》一书第 3.1 节，有：

1.

$$
\mathbf{H} = \begin{bmatrix}
1 & & & \\
-1 & 1 & & \\
 & -1 & 1 & \\
 & & -1 & 1 \\
1 & & & \\
 & 1 & & \\
 & & 1 & \\
 & & & 1
\end{bmatrix}
$$

2.

$$
\mathbf{W} = \begin{bmatrix}
\check{\mathbf{P}}_0 & & & & & & & \\
 & \boldsymbol{R}_1 & & & & & & \\
 & & \boldsymbol{R}_2 & & & & & \\
 & & & \boldsymbol{R}_3 & & & & \\
 & & & & \boldsymbol{Q}_0 & & & \\
 & & & & & \boldsymbol{Q}_1 & & \\
 & & & & & & \boldsymbol{Q}_2 & \\
 & & & & & & & \boldsymbol{Q}_3
\end{bmatrix}
$$

3. 存在唯一解，解为：

$$
(\mathbf{H}^{\mathrm{T}}\mathbf{W}^{-1}\mathbf{H})\hat{\mathbf{X}} = \mathbf{H}^{\mathrm{T}}\mathbf{W}^{-1}\mathbf{z}
$$