第四节课习题

高翔

2018 年 3 月 18 日

1 习题说明

- 第 i 节课习题所有材料打包在 Li.zip 中，$\forall i = 1 \dots 8$。
- 习题分为若干种：**计算类**习题，需要读者编程计算一个实际问题，我们会附有参考答案以供自测。**操作类**习题，会指导读者做一个具体的实验，给出中间步骤截图或结果。**简述类**习题则提供阅读材料，需要读者阅读材料后，回答若干问题。
- 每个习题会有一定的分值。每次习题分值加和为 10 分。你需要获得 8 分以上才能得到"通过"的评价。带 $*$ 的习题为附加题，会在总分之外再提供一定的分值，所以总和可能超过 10 分。换句话说，你也可以选择一道附加题，跳过一道正常题。
- 每道习题的给分由助教评判，简述类习题可能存在一定开放性，所以评分也存在主观因素。
- 请利用深蓝学院系统提交习题。每次习题我们会记通过与否。提交形式为 word 或 pdf 格式报告，如有编程习题请提交可编译的源码。
- 为方便读者，我通常会准备一些阅读材料，放在 books/或 papers/目录下。请读者按个人需求使用这些材料。它们多数是从网络下载的，如果侵犯到你的权利，请及时告诉我。
- 每个习题会标注大致用时，但视同学个人水平可能会有出入。
- 习题的完成情况会影响你对本课程内容的掌握程度，请认真、独立完成。**习题总得分较高的同学将获得推荐资格**。

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：

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

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。

注：本题不要使用 OpenCV 自带的去畸变函数，你需要自己理解去畸变的过程。我给你准备的程序中已经有了基本的提示。作为验证，去畸变后图像如图 2 所示。如你所见，直线应该是直的。

$$\left\{\begin{array}{l} u = f_x \cdot x + c_x \\ v = f_y \cdot y + c_y \end{array}\right.$$

$$\Rightarrow \left\{\begin{array}{l} x = \frac{1}{f_x}(u - c_x) \\ y = \frac{1}{f_y}(v - c_y) \end{array}\right.$$

$$\Downarrow$$

图 2: 验证图像

u, v 是像素坐标，单位是像素，是整数，每个像素点都含有图象信息（此处为灰度）
0~255 ⇒ unsigned char

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

$$r^2 = x^2 + y^2$$

$$\Downarrow$$

$$\left\{\begin{array}{l} u_d = f_x \cdot x_d + c_x \\ v_d = f_y \cdot y_d \cdot c_y \end{array}\right.$$

u_d, v_d 已知 但可能 ⟶ 不是整数，没有对应的像素点，所以对图象进行插值，再赋值给 u, v

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：

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

baseline

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。

$\frac{P \cdot m}{\frac{m}{P}}$ 1

→ disparity

$:= D$

$Z = \frac{fd}{D} = \frac{f d}{u_L - u_R} \cdot d.$

图 3: 双目图像的左图、右图与视差

$f \cdot \alpha$

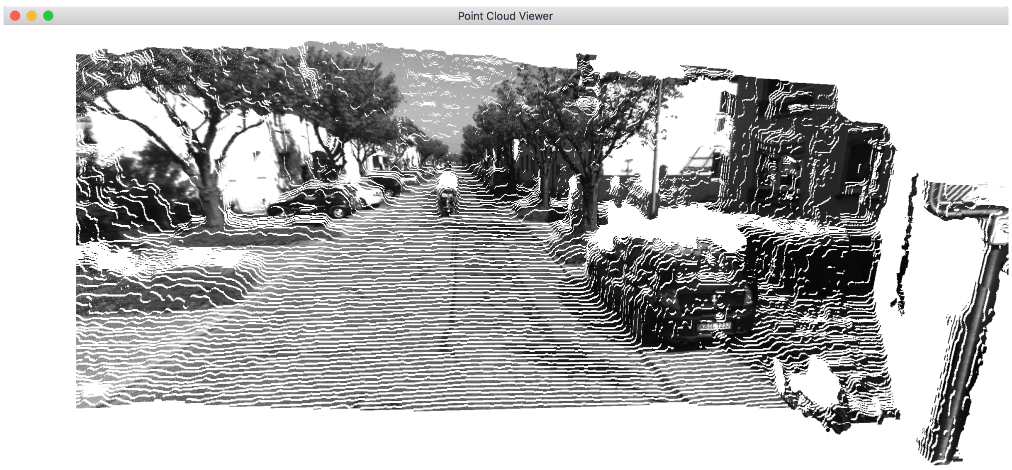
$f_x \cdot$

作为验证，生成点云应如图 4 所示。

$\frac{f}{u_L - u_R}$

Z

$x.$



$Z = \frac{fd}{D}$

图 4: 双目生成点云结果

$$\begin{cases} u_L = f_x \cdot \frac{X_L}{Z} + c_x \\ v = f_y \cdot \frac{Y}{Z} + c_y \end{cases}$$

$$\begin{cases} u_R = f_x \cdot \frac{X_R}{Z} + c_x \\ v = f_y \cdot \frac{Y}{Z} + c_y \end{cases}$$

$Z = \frac{fd}{D}$

$\underline{Z \cdot \alpha} = \frac{f_x d}{D}$

$m \cdot \frac{P}{m}$

⇓

pixel

x

$y.$

$tr(A) = \sum_i A_{ii}$

4 矩阵微分（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：

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

1. $\dfrac{d(Ax)}{dx} = \dfrac{dA}{dx}\cdot x + A\cdot\dfrac{dx}{x} = A$

2. $\dfrac{d(x^TAx)}{dx} = dx^TAx + x^TAdx = Ax + x^TA = Ax + A^Tx$
$= (A+A^T)x$

3. LHS: $d(xA^Tx) = (A^T + A)x$

RHS: $d[tr(Axx^T)] = Ax + A^Tx$

[1] 严格的写法必须对行向量求导，所以应该写成 $\mathrm{d}(\mathbf{Ax})/\mathrm{d}\mathbf{x}^\mathrm{T}$。但有些时候我们为了公式简洁，也会省略这个 $^\mathrm{T}$。

$f(x+\Delta x) \approx f(x) + J(x)\Delta x$

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。

$H\Delta x = b$

$LDL^T \Delta x = b$

$\frac{\partial e^{f(x)}}{\partial x} = e^{f(x)} \cdot f'(x)$

$\frac{\partial \exp(ax_i^2 + bx_i + c)}{\partial a} = \exp(ax_i^2 + bx_i + c) \cdot x_i^2$

$\frac{\partial e}{\partial b} = e \cdot x_i$

$\frac{\partial e}{\partial c} = e$

Bibliography

[1] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016.

[2] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," *2012 IEEE Conference On Computer Vision And Pattern Recognition (cvpr)*, pp. 3354–3361, 2012.

[3] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 807–814, IEEE, 2005.

[4] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1-3, pp. 7–42, 2002.