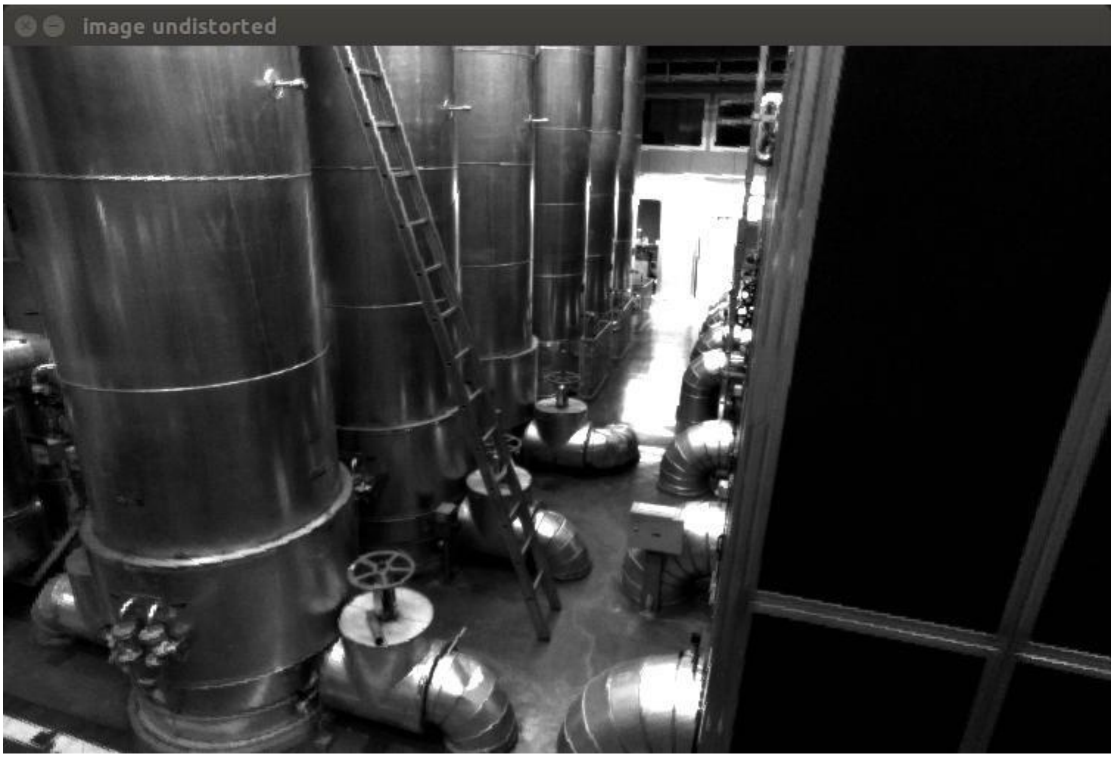
2 图像去畸变(3 分，约 1 小时)




```
// TODO 按照公式，计算点(u,v)对应到畸变图像中的坐标(u_distorted, v_distorted) (~6 lines)
// start your code here
double x = (u-cx)/fx;
double y = (v-cy)/fy;
double r = x * x + y * y;
u_distorted = x * (1 + k1 * r + k2 * r * r) + 2 * p1 * x * y + p2 * (r + 2 * x * x);
v_distorted = y * (1 + k1 * r + k2 * r * r) + p1 * (r + 2 * y * y) + 2 * p2 * x * y;
u_distorted = fx * u_distorted + cx;
v_distorted = fx * v_distorted + cy;
// end your code here
```

3 双目视差的使用(2 分，约 1 小时)


```
// start your code here (~6 lines)
// 根据双目模型计算 point 的位置
double x = (u - cx) /fx;
double y = (v - cy) /fy;
unsigned char disparity_value = disparity.at<uchar>(v,u);
double z = fx * d / disparity_value;
point[0] = x * z;
point[1] = y * z;
point[2] = z;
pointcloud.push_back(point);
// end your code here
```

4 矩阵运算微分(2 分，约 1.5 小时)

设变量为$x \in R^N$，那么:

1. 矩阵$A \in R^{NxN}$，那么$d(Ax)/dx$是什么？

$$Ax = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n \end{bmatrix} = \begin{bmatrix} (Ax)_1 \\ \vdots \\ (Ax)_n \end{bmatrix}$$

则$\frac{dAx}{dx} = \begin{bmatrix} \frac{d(Ax)_1}{dx} \\ \vdots \\ \frac{d(Ax)_n}{dx} \end{bmatrix} = \begin{bmatrix} \frac{d(Ax)_1}{dx_1} & \cdots & \frac{d(Ax)_1}{dx_n} \\ \vdots & \ddots & \vdots \\ \frac{d(Ax)_n}{dx_1} & \cdots & \frac{d(Ax)_n}{dx_n} \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$=A

2. 矩阵$A \in R^{NxN}$，那么$d(x^T Ax)/dx$是什么？

这等于于是一个标量对向量求导问题。其中$X^T AX = \sum_i^n \sum_j^n A_{ij} X_j X_i$

$$\frac{d(X^T AX)}{dX} = \begin{bmatrix} \frac{\partial \sum_i^n \sum_j^n A_{ij} X_j X_i}{\partial x_1} \\ \vdots \\ \frac{\partial \sum_i^n \sum_j^n A_{ij} X_j X_i}{\partial x_k} \\ \vdots \\ \frac{\partial \sum_i^n \sum_j^n A_{ij} X_j X_i}{\partial x_n} \end{bmatrix}$$

当一个变量多次出现时，可以分别对每次出现求导，再求和，则上式分别对x_j, x_i求导

$$= \begin{bmatrix} \sum_i^n A_{i1} X_i + \sum_j^n A_{1j} X_j \\ \vdots \\ \sum_i^n A_{ik} X_i + \sum_j^n A_{kj} X_j \\ \vdots \\ \sum_i^n A_{in} X_i + \sum_j^n A_{nj} X_j \end{bmatrix} = [AX \quad + \quad A^T X]$$

3. 证明: $x^T Ax = tr(Axx^T)$.

$$首先上题证明 d(X^T AX) = AX + A^T X$$

$$这里求 d\left(tr(AXX^T)\right) = tr\left(d(AXX^T)\right)$$

$$\begin{aligned} &= tr(dA * XX^T + A * dX * X^T + AX * dX^T) \\ &= tr(A * dX * X^T) + tr(AX * dX^T) \\ &= tr(X^T AdX) + tr((AX(dX)^T)^T) \\ &= tr(X^T AdX) + tr(X^T A^T dX) \\ &= tr((X^T A + X^T A^T) dX) \end{aligned}$$

$$= (X^T A + X^T A^T)^T$$
$$= A^T X + AX$$

5 高斯牛顿法的曲线拟合实验（3 分，约 2 小时）

```
xin@ubuntu16:~/VSLAM-course/vSLAM-course/Ch4/homework/SLAM[illegible]/L4/code/build$ ./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
total cost: 101.937
total cost: 101.937
cost: 101.937, last cost: 101.937
estimated abc = 0.890912, 2.1719, 0.943629
```

```
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;
//JJ^T * delta X = -J * f(x)
dx = H.ldlt().solve(b);
// end your code here
```

6 ∗ 批量最大似然估计(2 分，约 2 小时)

1. 可以定义矩阵 H，使得批量误差为 e = z - Hx。请给出此处 H 的具体形式。

在此系统中，v 为两帧之间的位移变量，x 为位移，而且 x 是直接观测。

我们的观测 Z 是$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$,则我们要有一个 H 矩阵把状态量$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$map(映射)到观测空间。

最小二乘问题就是找到使得映射结果最接近观测的状态量 X（映射结果和观测的误差最小)。而这种误差包括 v 的误差，即我们通过相邻帧$x_1 - x_0$的得到的 v 和实际 v 的误差:$\mathrm{error} = \mathrm{v}_{obser} - (x_k - x_{k-1})$以及我们通过前一帧预测的下一帧的位置$x_1$和直接观测到的$y_1$之间的误差:$\mathrm{error} = \mathrm{y_k} - x_k$。

$$则 e = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} - H \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, H = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. 请给出此问题下 W 的具体取值

因为最小二乘问题本质是求一个高斯分布的最大似然，而我们的观测和输入是互相独立的，那么他们的概率是乘积的形式合到一起。等于我们要求的”找到状态 X，使得到观测 z 和输入 u 的概率最大”的问题就可以分解成如下形式

$$\mathrm{argmaxP(z,u|x)} = \mathrm{argmaxP(z|x)} * P(u|x)$$

$$= \prod_{k=1}^{3} p(u_k|x_k, x_{k-1}) \prod_{k=1}^{3} p(u_k|x_k)$$

我们对高斯分布$\mathrm{P(z|x)}$和$P(u|x)$做概率密度展开，再取负对数，原最大概率问题变成了对下式求最小问题。

$$\mathrm{argmin} = \sum_{k=1}^{3} e_{v,k}^T Q_k^{-1} e_{v,k} + \sum_{k=1}^{3} e_{y,k}^T R_k^{-1} e_{y,k}$$

$$= \sum_{k=1}^{3} (v_k - x_k + x_{k-1})^T Q_k^{-1} (v_k - x_k + x_{k-1}) + \sum_{k=1}^{3} (y_k - x_k)^T R_k^{-1} (y_k - x_k)$$

第一题定义了$\mathrm{e = z - Hx} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} - \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$，可以得到$\mathrm{e} = \begin{bmatrix} \mathrm{v}_1 - x_1 + \mathrm{x}_0 \\ \mathrm{v}_2 - x_2 + \mathrm{x}_1 \\ \mathrm{v}_3 - x_3 + \mathrm{x}_2 \\ y_1 - x_1 \\ y_2 - x_2 \\ y_3 - x_3 \end{bmatrix}$

参考上面 $\mathrm{e^T Q_k e}$ 和$\mathrm{e^T R_k e}$ 的设计，易得 $\mathrm{W} = \begin{bmatrix} \mathrm{Q_k} & 0 \\ 0 & \mathrm{R_k} \end{bmatrix} = \begin{bmatrix} Q_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & Q_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & Q_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & R_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & R_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & R_3 \end{bmatrix}$

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。

有唯一解，需要 rank(H)>=size(X)。即公式数量超过未知数数量。这个问题其实就是 HX=Y 的最小二乘问题。

设$\mathrm{H} \in \mathrm{R}^{m*n}, X \in \mathrm{R}^{n*1}$

$$\mathrm{HX} = \mathrm{Y}$$

$$f(x) = Y - HX$$

$$minF(x) = \frac{1}{2}\|f(x)\|^2$$

对 X 求导并令导数=0，有

$$\frac{\mathrm{d}minF(x)}{\mathrm{dx}} = \frac{d\frac{1}{2}(\mathrm{Y}-\mathrm{HX})^2}{\mathrm{d}(\mathrm{Y}-\mathrm{HX})} * \frac{d(\mathrm{Y}-\mathrm{HX})}{dX} = 0$$

$$= (Y - HX) * (-H) = 0$$

$$H^T HX = \mathrm{H}^{\mathrm{T}} Y$$

$$X = (H^T H)^{-1} H^T Y$$