本程序采用最小二乘法进行直线拟合，通过计算点与拟合直线的距离进行离群点剔除。
具体流程结合代码进行展示：

```
float k = 0.7, b = 5;
float k_find = 0, b_find = 0;

RNG rng;
vector<Point> point;
Mat img = Mat(1000, 1000, CV_8UC1);
Mat img2 = Mat(1000, 1000, CV_8UC1);
img = Scalar(0);
img2 = Scalar(0);
for (int i = 0; i < 15; i++)
{
    float x = rng.uniform(0, 1000);
    float y = k*x +b+ rng.gaussian(10);
    if (i == 7) y += 200;
    if (i == 8) y += 500;
    //每随机生成一个Point就把它存入vector中
    point.push_back(Point(x, y));
    //显示
    circle(img, point[i], 5, (0, 0, 255), 0);
}
```

这部分代码用于生成一组带噪音的点，并把第 7 和第 8 个点的 y 值增加 200 和 500 人为设置成离群点。

```
void Plot_img(Mat &img, vector<Point> &point, float& k, float& b)
{
    vector<float> x;
    vector<float> y;

    for (int i = 0; i < point.size(); i++)
    {
        x.push_back(point[i].x);
        y.push_back(point[i].y);
    }
    float t1 = 0, t2 = 0, t3 = 0, t4 = 0;
    for (unsigned int i = 0; i < x.size(); i++)
    {
        t1 += x[i] * x[i];
        t2 += x[i];
        t3 += x[i] * y[i];
        t4 += y[i];
    }
    k = (t3 * x.size() - t2 * t4) / (t1 * x.size() - t2 * t2);
    b = (t1 * t4 - t2 * t3) / (t1 * x.size() - t2 * t2);
    cout << k << endl;
    cout << b << endl;
}
```

这部分代码用于生成拟合直线，采用最小二乘法，k 和 b 的计算采用的是最小二乘法的公式。

直线拟合后未剔除离群点的拟合直线如图 1：

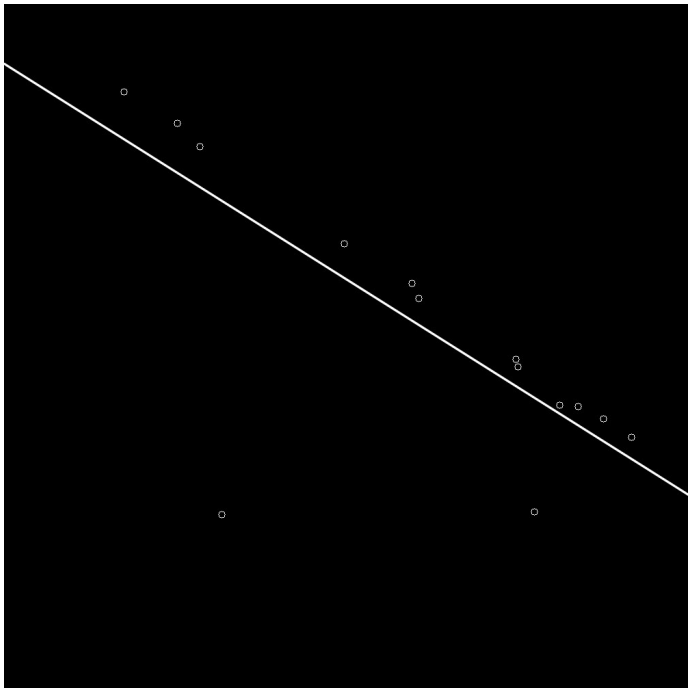

图 1. 拟合直线未剔除离群点

```
vector<Point> mad(vector<Point> &point, float s, float& k, float& b)
{
    int n = point.size();
    if (n < 2)
    {
        return point;
    }
    vector<float> distance;
    vector<float> distance2;
    float x1 = 0, y1 = b, x2 = 100, y2 = 100 * k + b;
    for (int i = 0; i < n; i++)
    {
        distance.push_back(fabs(((y1 - y2) * point[i].x - (x1 - x2) * point[i].y + (x1 * y2 - x2 * y1)) / sqrt(pow(y1 - y2, 2) + pow(x1 - x2, 2))));
    }
    distance2 = distance;
    sort(distance.begin(), distance.end());

    float median = (n % 2 == 1) ? distance[n / 2] :
        (distance[n / 2] + distance[n / 2 - 1]) / 2;//中位数
    vector<float> deviations;
    vector<Point> new_nums;     //偏差值
    for (int i = 0; i < n; i++)
    {
        deviations.push_back(abs(distance[i] - median));
    }

    sort(deviations.begin(), deviations.end());
    float mad = (n % 2 == 1) ? deviations[n / 2] :
        (deviations[n / 2] + deviations[n / 2 - 1]) / 2;
    for (size_t i = 0; i < n; i++)
    {
        if (abs(distance2[i] - median) < s * mad)
            new_nums.push_back(point[i]);
    }
    return new_nums;
}
```

这部分代码用于剔除离群点，主要方法是通过：

1、计算所有点分别与所拟合直线的距离

2、对距离进行排序，找出中间值

3、计算每个距离与距离中位数的偏差值

4、对偏差值进行排序，找出偏差值的中位数作为基准

5、将所有偏差值与这个中位数缩放后的数值进行比较，其中 s 是缩放因子，用于调整可接受的离群值偏差范围，例如，s=1 即为接受偏差最小的一半数据。如图 2 为 s=1，图 3 为 s=2。

重新拟合直线即可获得剔除离群点的直线。

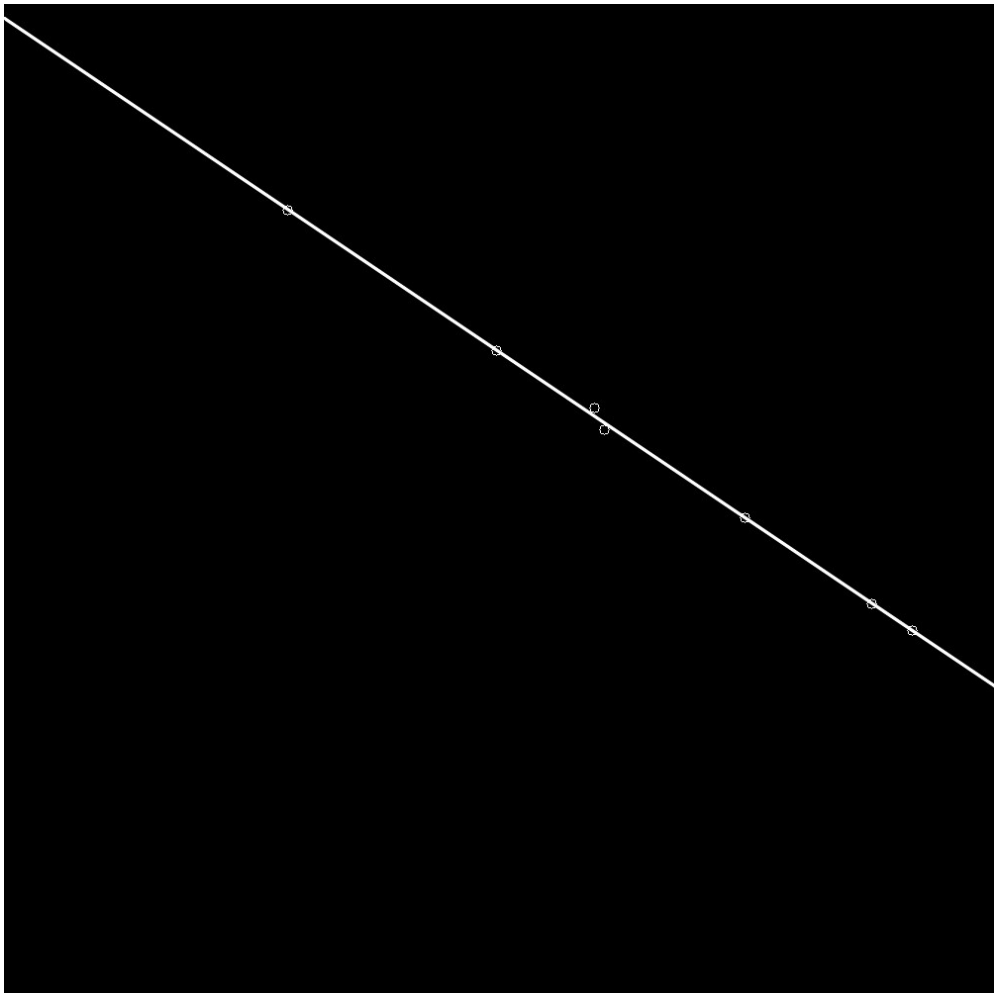

图 2 剔除离群点拟合直线 s=1

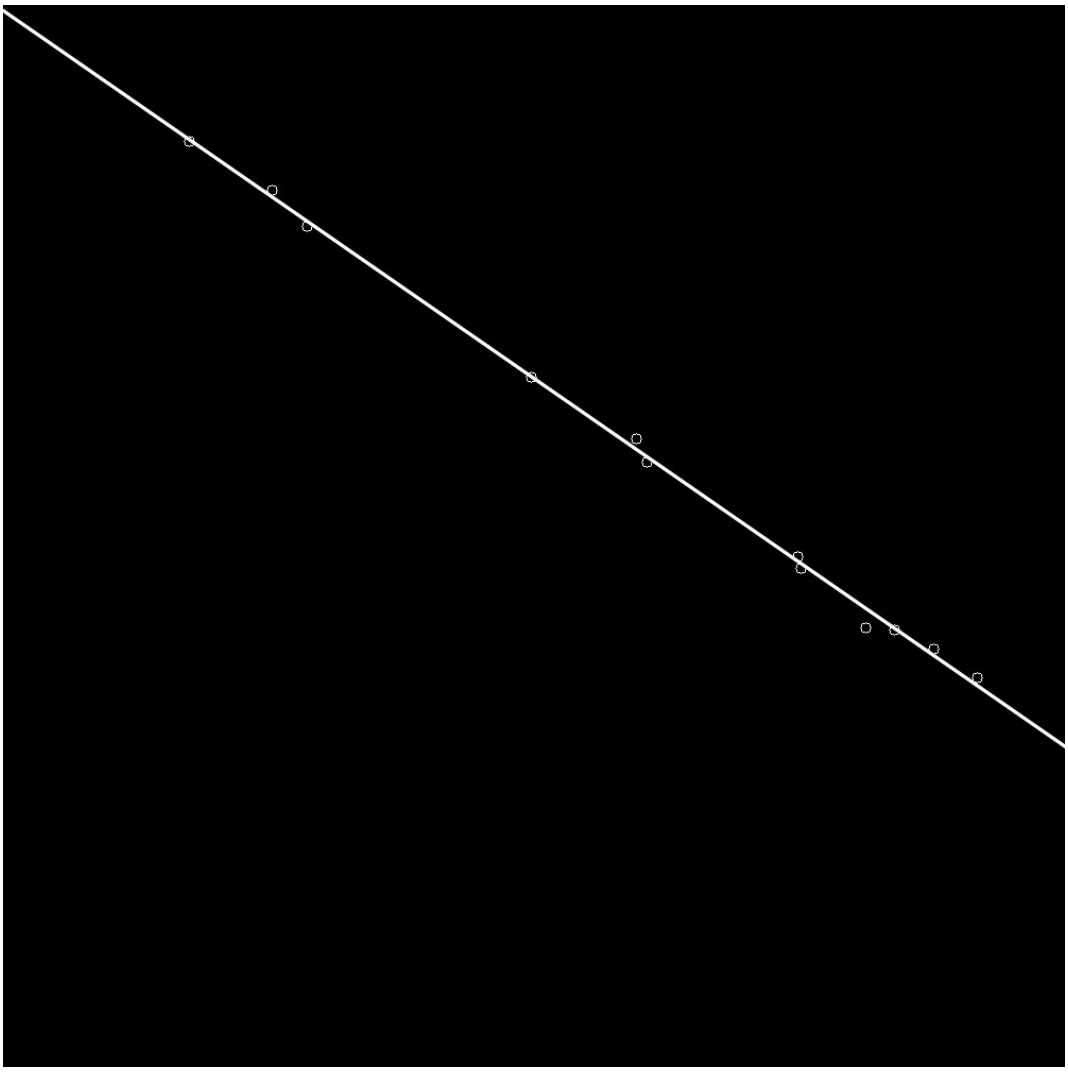

图 3 剔除离群点拟合直线 s=2