相机位姿估计的加速正交迭代算法

李 鑫 龙古灿 刘进博 张小虎 于起峰

[1]国防科学技术大学航天科学与工程学院, 湖南 长沙 410073

[2]国防科学技术大学图像测量与视觉导航湖南省重点实验室, 湖南 长沙 410073

摘要 面对需要实时计算的相机位姿估计问题,针对经典的广泛应用的正交迭代算法,提出了一种加速正交迭代算法。其关键思想是将每一次迭代过程规整化,从而提炼出每一次迭代的重复计算,若将此重复计算在迭代开始前提前计算,则可以大幅度的减少迭代过程中的计算量,使得每一次迭代的计算复杂度从O(n)降低为O(1)。因此,可以在更短的时间内迭代更多的次数,从而获得更高的精度。进行了对比实验,结果显示本加速算法计算精度更高,速度更快。并通过实验提出了选择稳健n点透视(RPnP)计算初值,再使用加速正交迭代算法进行迭代运算的方法,在控制点不多的情况下,是一种精度接近最大似然估计,计算速度最快的算法。

关键词 机器视觉;相机位姿估计;加速正交迭代;计算复杂度;最大似然估计

中图分类号 O436 **文献标识码** A

doi: 10.3788/AOS201535.0115004

Accelerative Orthogonal Iteration Algorithm for Camera Pose Estimation

Li Xin Long Gucan Liu Jinbo Zhang Xiaohu Yu Qifeng

[1]*College of Aerospace Science and Engineering, National University of Defense Technology, Changsha, Hunan* 410073, *China*

[2]*Hunan Key Laboratory of Videometrics and Vision Navigation, National University of Defense Technology, Changsha, Hunan* 410073, *China*

Abstract An accelerative orthogonal iteration algorithm about the classical and wildly used orthogonal iteration algorithm for camera pose estimation is proposed for real time computation. The key idea is to integrate the steps in each iteration. The repetitive computation in each iteration can be abstracted and done before iteration. The computational complexity of each iteration is reduced from O(n) to O(1). So that, more iteration can be done in short time, and the accuracy is improved as well. The contrastive simulation and real data experiments show the efficiency and accuracy of the accelerative algorithm. Experimentally, the accelerative algorithm with the robust perspective n point (RPnP) initialization has nearly the same accuracy as maximum likelihood estimation (MLE), and is the fastest algorithm when there are few control points.

Key words machine vision; camera pose estimation; accelerative orthogonal iteration; computational complexity; maximum likelihood estimation

OCIS codes 150.1488; 150.6910; 120.4640

1 引 言

相机位姿估计问题是指在相机内参已知的情况下,给定空间点及其在图像上的像点坐标,求解相机位置和姿态的问题,通常称为n点透视(PnP)问题。这是计算机视觉的经典问题,在计算机视觉[1]、摄影测量[2]、机器人学[3]和增强现实[4]方面有很多重要的应用。

收稿日期: 2014-07-08; **收到修改稿日期**: 2014-09-02

基金项目: 国家自然科学基金(11072263,11332012)

作者简介: 李鑫(1985—),男,博士研究生,主要从事大型结构变形测量及位姿估计、计算机视觉等方面的研究。E-mail: lixin_nudt@163.com

导师简介: 于起峰(1958—),男,教授,博士生导师,中国科学院院士,主要从事空天图像测量与视觉导航方面的研究。E-mail: yuqifeng@vip.sina.com

求解PnP问题的最少点数为3，P3P最多有4个解[5-6]。通常，求解PnP问题的算法按照是否迭代分为两类：线性算法和非线性迭代算法。较早的线性算法计算复杂度都较高，如Quan[7]($n\geqslant 4$)为O(n^5)，Fiore[8]($n\geqslant 6$)为$O(n^2)$，Ansar[9]($n\geqslant 4$)为O(n^8)。计算复杂度为O(n)的线性算法主要包括直接线性变换(DLT)[1]($n\geqslant 6$)、EPnP[10]($n\geqslant 4$)、稳健PnP(RPnP)[11]($n\geqslant 4$)、直接最小二乘法(DLS)[12]($n\geqslant 3$)、OPnP[13]($n\geqslant 3$)。其中前三者计算速度较快，但一般得不到较高的精度。而后两者具有明确的目标函数，所以精度较高，使用了代数多元多次方程组求解技术，虽然能达到全局最优，但计算速度较慢。

非线性算法精度一般较高，且由于只输出一个解，因此一般要求 $n\geqslant 4$ 。POSIT[14]算法反复的线性计算有尺度的正射投影相机的位姿，采用轮流地计算位姿和与深度相关的比例因子的策略。正交迭代算法[15]建立了物方残差平方和目标函数，不断地使用绝对定向进行迭代。文献[16]则将其推广到多摄像机系统。gOp[17]方法也以物方残差平方和为目标函数，将问题转化为半正定规划问题(SDP)，从而通过SDP求解相机位姿估计问题。PPnP[18]方法也是最小化一种变换的物方残差平方和目标函数，建立各向异性强迫模型，不断的迭代更新世界点的深度，通过奇异值分解(SVD)确定旋转矩阵。Hartley等[19]利用遍历搜索空间和分支定界方法寻找无穷范数的全局最优解，计算量巨大。非线性算法除了需要初值，最重要的一点就是计算量较大，计算时间较长。

正交迭代算法是一种经典的广泛使用的迭代算法，虽然相比其他迭代算法，速度较快，但面对一些需要实时计算的情况时，除了精度要求外，计算时间也有要求。

提出了一种正交迭代加速算法，关键思想是在正交迭代算法的基础上，将每一次迭代过程规整化，提炼出每一次迭代的重复计算，将此重复计算在迭代开始前计算，从而可以大大减少迭代过程中的计算量。此方法可以使得每一次迭代过程的复杂度从O(n)降低为O(1)，而在迭代前需要提前计算的复杂度仍为O(n)。

2 正交迭代加速算法

2.1 正交迭代算法

不同于最小化像方残差，正交迭代算法[15]的目标是为了最小化物方残差。图1给出了像方残差与物方残差的几何意义。物方残差目标函数为

$$E(\boldsymbol{R},\boldsymbol{t})=\sum_{i=1}^{n}\left\|(\boldsymbol{I}-\hat{\boldsymbol{V}}_i)(\boldsymbol{R}\boldsymbol{p}_i+\boldsymbol{t})\right\|^2, \tag{1}$$

式中 $\boldsymbol{I}$ 为单位矩阵，$\hat{\boldsymbol{V}}_i=\hat{\boldsymbol{v}}_i\hat{\boldsymbol{v}}_i^{\mathrm{T}}/(\hat{\boldsymbol{v}}_i^{\mathrm{T}}\hat{\boldsymbol{v}}_i)$ 为视线投影矩阵，$\hat{\boldsymbol{v}}_i=[u_i \quad v_i \quad 1]^{\mathrm{T}}$ 为控制点投影到归一化像面上的像点坐标，$\boldsymbol{p}_i$ 为控制点坐标，$\boldsymbol{R},\boldsymbol{t}$ 为相机在控制点坐标系下的位姿。

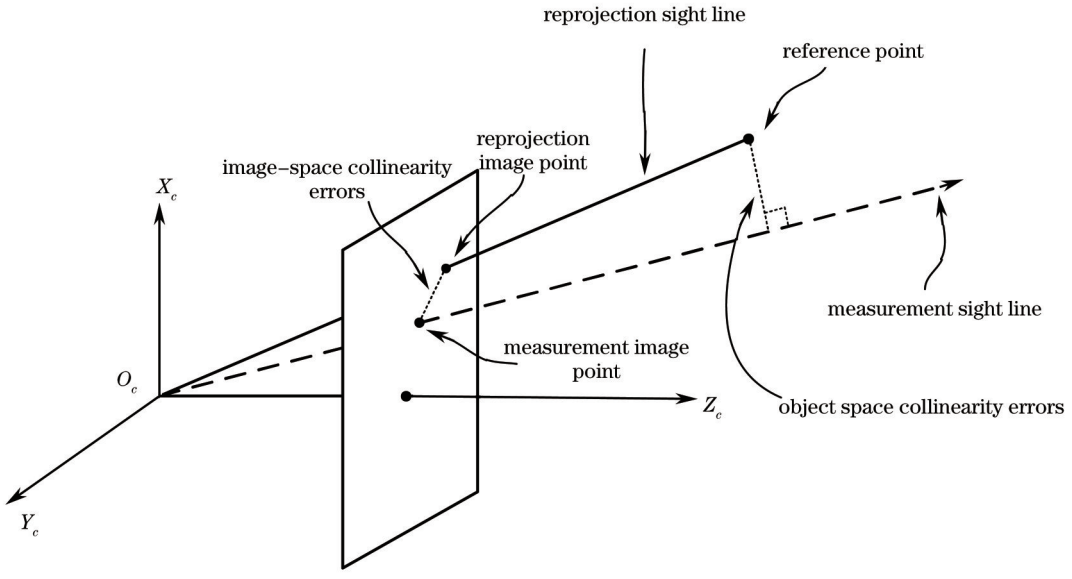

图1 物方残差与像方残差的几何意义

Fig.1 Geometric meanings of object-space and image-space collinearity errors

而目标函数(1)在已知旋转矩阵 $\boldsymbol{R}$ 的情况下，平移向量 $\boldsymbol{t}$ 存在最优解，为

$$\boldsymbol{t}(\boldsymbol{R})=\frac{1}{n}\left(\boldsymbol{I}-\frac{1}{n}\sum_{i=1}^{n}\hat{\boldsymbol{V}}_i\right)^{-1}\sum_{i=1}^{n}(\hat{\boldsymbol{V}}_i-\boldsymbol{I})\boldsymbol{R}\boldsymbol{p}_i. \tag{2}$$

其迭代过程开始时需要给一个初始的 $\boldsymbol{R}$，然后再求出最优 $\boldsymbol{t}$，再利用基于SVD的绝对定向问题的最优解[20]更新 $\boldsymbol{R}$，从而不断迭代更新 $\boldsymbol{R}$ 和 $\boldsymbol{t}$。在已知第 k 次迭代的 $\boldsymbol{R}^k$，$\boldsymbol{t}^k$ 后，通过求解下式的绝对定向问题，获得 $k+1$ 次迭代的 $\boldsymbol{R}$。

$$\boldsymbol{R}^{k+1}=\arg\min_{\boldsymbol{R}}\sum_{i=1}^{n}\left\|\boldsymbol{R}\boldsymbol{p}_i+\boldsymbol{t}-\boldsymbol{o}_i^k\right\|^2, \tag{3}$$

式中 $\boldsymbol{o}_i^k=\hat{\boldsymbol{V}}_i\boldsymbol{q}_i^k$ 表示点 $\boldsymbol{q}_i^k$ 在对应视线上的投影点，而 $\boldsymbol{q}_i^k=\boldsymbol{R}^k\boldsymbol{p}_i+\boldsymbol{t}^k$ 表示第 k 次迭代后，参考点经过当前的 $\boldsymbol{R}^k$，$\boldsymbol{t}^k$ 变换后，在相机坐标系下的坐标。

迭代停止的条件选择为目标函数小到一定程度，或者目标函数的相对变化量小到一定程度，或者达到了预设的迭代次数上限。

2.2 加速正交迭代算法

提出的加速正交迭代算法主要是在原算法的基础上在下面3个方面进行规整：

1) $\boldsymbol{t}$ 的规整。正交迭代算法每一次的迭代过程需要分别计算一次 $\boldsymbol{R}$ 和 $\boldsymbol{t}$。实际上，其本质是对 $\boldsymbol{R}$ 的迭代，因为每一次更新 $\boldsymbol{R}$ 后，都可以线性计算出最优 $\boldsymbol{t}$。$\boldsymbol{t}$ 的求解除了最后一次迭代需要输出结果，其他迭代的时候只是一个中间值。因此，在迭代的中间过程可以通过整合，消除 $\boldsymbol{t}$ 的求解；

2) 投影点的规整。原算法在每一次的迭代过程中都需要重新计算投影点，这些投影点的计算只是为了利用绝对定向算法更新 $\boldsymbol{R}$，本质上投影点的计算也只是一个中间过程。因此，在迭代过程中也可以通过整合消除投影点的求解；

3) 目标函数的规整。目标函数在每一次迭代过程中都要计算，而且是 $\boldsymbol{R}$ 和 $\boldsymbol{t}$ 的函数。但是，由于在已知 $\boldsymbol{R}$ 的情况下，$\boldsymbol{t}$ 存在线性解，因此，本质上目标函数只是 $\boldsymbol{R}$ 的函数。并且容易发现，目标函数是 $\boldsymbol{R}$ 的二次函数。

为了方便后续的推导，首先引入一个矩阵计算的公式：

$$\mathrm{vec}(\boldsymbol{ABC})=(\boldsymbol{C}^{\mathrm{T}}\otimes\boldsymbol{A})\mathrm{vec}(\boldsymbol{B}), \tag{4}$$

式中 vec(·) 表示把一个矩阵按列堆栈成一个列向量，$\otimes$ 表示 Kronecker 积。

对参考点进行零均值化

$$\boldsymbol{p}_i\leftarrow\boldsymbol{p}_i-\bar{\boldsymbol{p}}, \tag{5}$$

式中 $\bar{\boldsymbol{p}}$ 表示所有参考点的平均值。

根据(2)，利用公式(4)，有

$$\boldsymbol{t}=\boldsymbol{G}_{3\times9}\boldsymbol{r}, \tag{6}$$

式中 $\boldsymbol{r}=\mathrm{vec}(\boldsymbol{R})$，

$$\boldsymbol{G}=\frac{1}{n}\left(\boldsymbol{I}-\frac{1}{n}\sum_{j=1}^{n}\hat{\boldsymbol{V}}_j\right)^{-1}\sum_{j=1}^{n}[\boldsymbol{p}_j^{\mathrm{T}}\otimes(\hat{\boldsymbol{V}}_j-\boldsymbol{I})]=\frac{1}{n}\left(\boldsymbol{I}-\frac{1}{n}\sum_{j=1}^{n}\hat{\boldsymbol{V}}_j\right)^{-1}\sum_{j=1}^{n}(\boldsymbol{p}_j^{\mathrm{T}}\otimes\hat{\boldsymbol{V}}_j). \tag{7}$$

对投影点进行规整：

$$\boldsymbol{o}_i^k=\hat{\boldsymbol{V}}_i\boldsymbol{q}_i^{(k)}=\hat{\boldsymbol{V}}_i(\boldsymbol{R}^k\boldsymbol{p}_i+\boldsymbol{t}^k)=(\boldsymbol{p}_i^{T}\otimes\hat{\boldsymbol{V}}_i+\hat{\boldsymbol{V}}_i\boldsymbol{G})\boldsymbol{r}^k=\boldsymbol{J}_i\boldsymbol{r}^k. \tag{8}$$

为了利用绝对定向最优解进行迭代，在每一迭代中需要计算矩阵

$$\boldsymbol{M}^k=\sum_{i=1}^{n}(\boldsymbol{o}_i^k-\bar{\boldsymbol{o}}^k)\boldsymbol{p}_i^{\mathrm{T}}=\sum_{i=1}^{n}\boldsymbol{o}_i^k\boldsymbol{p}_i^{\mathrm{T}}=\sum_{i=1}^{n}\boldsymbol{J}_i\boldsymbol{r}^k\boldsymbol{p}_i^{\mathrm{T}}, \tag{9}$$

式中 $\bar{\boldsymbol{o}}^k$ 表示第 k 次迭代时所有投影点的平均值。设 $\boldsymbol{m}^k=\mathrm{vec}(\boldsymbol{M}^k)$，则根据(9)式，有

$$\boldsymbol{m}^k=\sum_{i=1}^{n}(\boldsymbol{p}_i\otimes\boldsymbol{J}_i)\boldsymbol{r}^k=\boldsymbol{B}_{9\times9}\boldsymbol{r}^k, \tag{10}$$

从而确定了 $\boldsymbol{M}^k$。根据绝对定向最优解，对其进行奇异值分解有 $\boldsymbol{M}^k=\boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\mathrm{T}}$，则

$$\boldsymbol{R}^{k+1}=\boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}, \tag{11}$$

(11)式给出了正交迭代算法更新 $\boldsymbol{R}$ 矩阵的表达式，而(10)式中，$\boldsymbol{B}$ 矩阵可以用(12)式在迭代开始之前求解出来。并且迭代过程中 $\boldsymbol{B}$ 矩阵是常矩阵。

$$\begin{aligned}\boldsymbol{B}=\sum_{i=1}^{n}\left(\boldsymbol{p}_i\otimes\boldsymbol{J}_i\right)=\sum_{i=1}^{n}\left[\boldsymbol{p}_i\otimes\left(\boldsymbol{p}_i^{\mathrm{T}}\otimes\hat{\boldsymbol{V}}_i+\hat{\boldsymbol{V}}_i\boldsymbol{G}\right)\right]=\sum_{i=1}^{n}\left(\boldsymbol{p}_i\otimes\boldsymbol{p}_i^{\mathrm{T}}\otimes\hat{\boldsymbol{V}}_i\right)+\sum_{i=1}^{n}\left[\boldsymbol{p}_i\otimes\left(\hat{\boldsymbol{V}}_i\boldsymbol{G}\right)\right]=\sum_{i=1}^{n}\left(\boldsymbol{p}_i\otimes\boldsymbol{p}_i^{\mathrm{T}}\otimes\hat{\boldsymbol{V}}_i\right)+\\\sum_{i=1}^{n}\left(\boldsymbol{p}_i\otimes\hat{\boldsymbol{V}}_i\right)(1\otimes\boldsymbol{G})=\sum_{i=1}^{n}\left(\boldsymbol{p}_i\otimes\boldsymbol{p}_i^{\mathrm{T}}\otimes\hat{\boldsymbol{V}}_i\right)+\sum_{i=1}^{n}\left(\boldsymbol{p}_i\otimes\hat{\boldsymbol{V}}_i\right)(\boldsymbol{G}).\end{aligned} \tag{12}$$

在迭代过程中，只需要存储 $\boldsymbol{B}$ 矩阵，就可以利用(10)、(11)式，对 $\boldsymbol{R}$ 进行迭代计算，迭代结束时输出 $\boldsymbol{R}_{\mathrm{out}}$。最后再利用(6)式计算一次 $\boldsymbol{t}_{out}$。因为一开始对参考点进行了零均值化，所以最后再对 $\boldsymbol{t}$ 进行平移。最终输出 $\boldsymbol{t}$ 的公式为

$$\boldsymbol{t}_{\mathrm{out}}\leftarrow\boldsymbol{t}_{\mathrm{out}}-\boldsymbol{R}_{\mathrm{out}}\bar{\boldsymbol{p}}. \tag{13}$$

很容易看出来，根据(10)、(11)式，加速后，每一次迭代过程中，求解 $\boldsymbol{R}$ 的计算复杂度为O(1)，也就是常值。

针对目标函数(1)，有

$$E(\boldsymbol{R},\boldsymbol{t})=\sum_{i=1}^{n}\left\|(\boldsymbol{I}-\hat{\boldsymbol{V}}_i)(\boldsymbol{R}\boldsymbol{p}_i+\boldsymbol{t})\right\|^2=\sum_{i=1}^{n}\left\|(\boldsymbol{I}-\hat{\boldsymbol{V}}_i)(\boldsymbol{p}_i^{\mathrm{T}}\otimes\boldsymbol{I}+\boldsymbol{G})\boldsymbol{r}\right\|^2=\boldsymbol{r}^{\mathrm{T}}\boldsymbol{C}_{9\times9}\boldsymbol{r}, \tag{14}$$

式中

$$\boldsymbol{C}=\sum_{i=1}^{n}(\boldsymbol{p}_i\otimes\boldsymbol{I}+\boldsymbol{G}^{\mathrm{T}})(\boldsymbol{I}-\hat{\boldsymbol{V}}_i)(\boldsymbol{p}_i^{\mathrm{T}}\otimes\boldsymbol{I}+\boldsymbol{G}), \tag{15}$$

式中 $\boldsymbol{p}_i$ 用的是零均值化后的 $\boldsymbol{p}_i$。可见，$\boldsymbol{C}$ 矩阵也可以在迭代开始前计算，在迭代过程中，$\boldsymbol{C}$ 矩阵保持不变，从而根据(14)式，每一次目标函数的计算复杂度也为O(1)。

综合以上所述，只要在迭代开始前，先计算 $\boldsymbol{G}$，然后再计算 $\boldsymbol{B}$ 和 $\boldsymbol{C}$ 矩阵。迭代过程中仅仅使用 $\boldsymbol{B}$ 和 $\boldsymbol{C}$ 矩阵，利用(10)、(11)式更新旋转矩阵 $\boldsymbol{R}$，利用(14)式计算目标函数，所以每一次迭代的计算复杂度就为O(1)，从而可以大幅度减少迭代过程中的计算量。迭代结束之后，再利用(6)、(13)式计算平移向量 $\boldsymbol{t}$ 即可。通过观察也可以发现，需要计算 $\boldsymbol{G}$、$\boldsymbol{B}$ 和 $\boldsymbol{C}$ 的(7)、(12)和(15)式的计算复杂度均为O(n)。

3 实 验

3.1 仿真实验

为了验证提出的正交迭代加速算法的有效性，在计算精度和计算时间上，将其与原正交迭代算法和其他的几种算法进行比较。

仿真中选择的默认参数为：

相机的内参矩阵

$$\boldsymbol{K}=\begin{bmatrix}800 & & 640\\ & 800 & 480\\ & & 1\end{bmatrix}$$

控制点在相机坐标系下的坐标均匀分布在[−2,2]×[−2,2]×[4,8]内，平移向量选择为这些控制点的中心，也就是将世界坐标系的原点建立在控制点的中心，这样正好可以使得控制点的世界坐标已经零均值化，旋转矩阵为随机三维(3D)旋转矩阵。像点加入的噪声水平为1 pixel。定义旋转的计算误差为：$e_{\mathrm{rot}}=\max_{k=1}^{3}\arccos[\mathrm{dot}(\boldsymbol{r}_{\mathrm{true}}^{k},\boldsymbol{r}^{k})]\times180/\pi$，其中 $\boldsymbol{r}_{\mathrm{true}}^{k},\boldsymbol{r}^{k}$ 分别为 $\boldsymbol{R}_{true},\boldsymbol{R}$ 的第 k 列。平移的计算误差为：$e_{\mathrm{trans}}(\%)=\left\|\boldsymbol{t}_{\mathrm{true}}-\boldsymbol{t}\right\|/\left\|\boldsymbol{t}_{\mathrm{true}}\right\|\times100$。

比较下列算法的精度与计算时间：

1) EPnP+GN. 文献[10]中的高效EPnP加上少许的几步高斯牛顿迭代；

2) RPnP. 文献[11]中的稳健位姿估计方法；

3) DLS. 文献[12]中的直接最小二乘方法；

4) DLS+++. 文献[12]中为了解决Cayley变换奇异的情况，提出的分别计算3次DLS的修正方法；

5) OP*n*P. 文献[13]中的全局最优解；

6) PP*n*P. 文献[18]中提出的各向异性强迫模型迭代解；

7,) LHM. 文献[15]中的正交迭代算法，以弱透视为初值，迭代次数上限为20；

8) LHM+.提出的加速正交迭代算法，由于每一次迭代的复杂度为O(1)。因此将迭代次数的上限设为100，仍然以弱透视为初值；

9) RP*n*P+LHM. 以RP*n*P为初值的正交迭代算法；

10) RP*n*P+LHM+. 以RP*n*P为初值的加速正交迭代算法；

11) RP*n*P+LM. 以RP*n*P为初值的最小化像方残差的Levenberg-Marquardt算法。

图2给出了在控制点数从4个增加到15个，不同算法的误差统计情况。第一列为旋转矩阵误差，第二列为平移向量误差，第一行为误差平均，第二行为误差中值。图3给出了不同点数下，不同算法在现有一般配置PC机下使用Matlab软件实现的平均计算时间的统计结果。

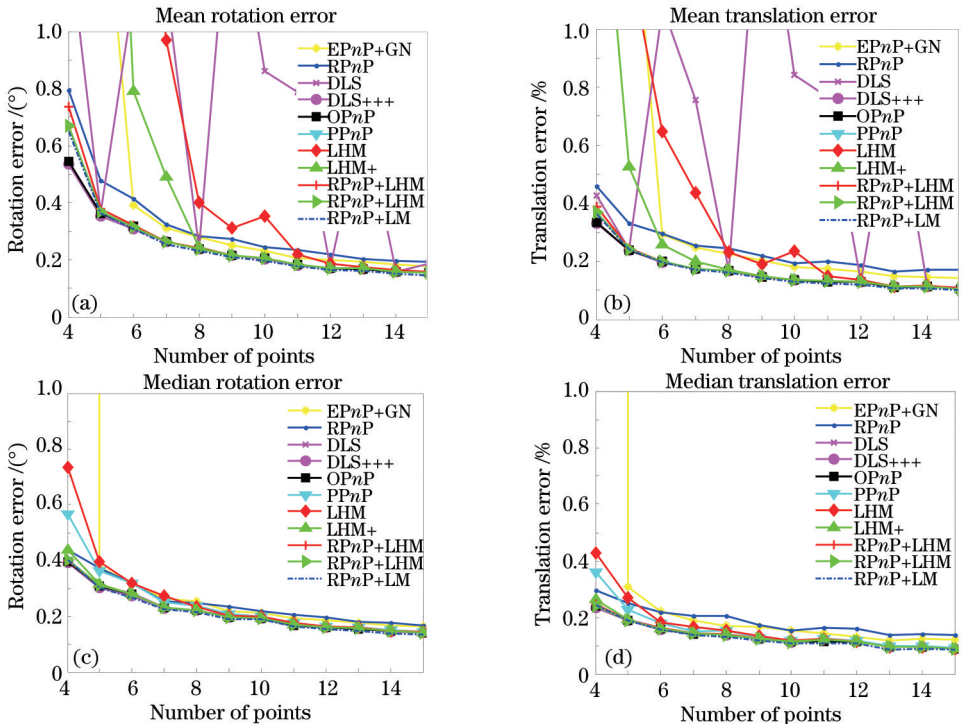

图2 不同算法随着控制点数的增多的误差对比。(每一个点都进行了500次独立实验)

Fig.2 Mean and median errors of rotation and translation with varying point numbers. (Each point in the plot represents 500 trials)

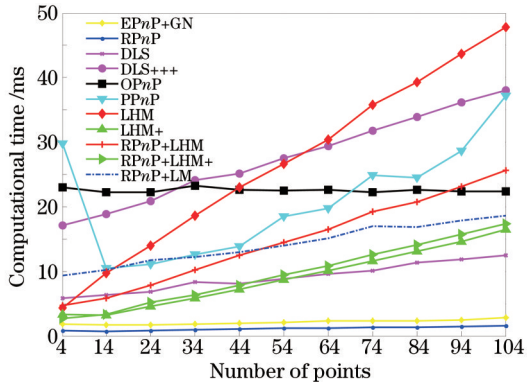

图3 不同算法随着点数的增加计算时间的结果。(每一点表示500次独立仿真实验的平均计算时间)

Fig.3 Average running time with varying point numbers. (Each point in the plot represents 500 trials)

从以上的仿真实验结果图中可以得到：

1) LHM+的精度高于LHM，且计算时间明显少于LHM。因为LHM+迭代次数的上限从20改为100次，从而可以迭代更多的次数，使得目标函数更小，精度更高。LHM+的每一次迭代的计算复杂度为O(1)，因此，即使迭代了更多的次数，计算时间仍然较少；

2) 利用RP*n*P作为初值的LHM和LHM+，精度有显著提高，特别是在点少（$n<12$）的情况下。主要原因是弱透视的精度不高，更容易使得LHM或LHM+收敛到局部最优解；

3) RP*n*P+LHM+的计算时间比RP*n*P+LHM的少；

4) RP*n*P+LHM相比于LHM，计算时间大幅减少；

5) RP*n*P+LHM+相比于LHM+，点多的时候，计算时间有所增加，因为减少少量的迭代次数对LHM+的计算时间影响不大。而RP*n*P是在点少时，精度较高，速度最快的线性算法，因此，点少时，使用RP*n*P作为初值，能够大幅度减少迭代次数，从而使得RP*n*P+LHM+的计算时间较少；

6) 4个点的时候，RP*n*P+LHM+的精度略低于OP*n*P和DLS+++。因为后两者均为全局优化方法，而在4个点的时候最容易出现共面的情况，有可能使得RP*n*P+LHM+收敛致局部最优，但RP*n*P+LHM+需要的时间明显比后两者少很多；

7) 精度比较接近且最高的几种算法包括：OP*n*P、DLS+++、RP*n*P+LHM、RP*n*P+LHM+、RP*n*P+LM。其中计算速度最快的是RP*n*P+LHM+。

对比中值和均值误差，可以看出，有些算法的均值误差很大，原因是，这些算法存在奇异情况（DLS），或迭代收敛致局部最优（LHM、LHM+、PP*n*P）的情况。PP*n*P在点少的时候需要的计算时间更多，是因为其采用的是固定初值，且在点少的时候收敛的较慢。

从而可以看出，提出的加速算法具有明显的加速效果，且由于短时间可以迭代更多的次数，使得精度也得到了提高。若再采用RP*n*P作为初值，精度高的同时速度也快。

3.2 真实实验

为了验证提出的加速正交迭代算法的有效性，进行一组真实实验。世界控制点采用9个红外十字标识灯，固定在一个边长约为20 cm的立方体形的架子上。其中4个控制点在向前的面上的4个角上，其他5个分布在向后的面上。具体分布结构如图4所示。相机镜头前安装红外滤光片，从而可以清晰的只成像出红外十字标识灯的像。相机的内参提前使用Zhang[21]的方法标定。图5表示一组成像及测量结果，白色较大的十字表示对标志灯成的像，红色十字表示提取的十字中心像点，绿色叉表示利用LHM+进行位姿结算后的重投影像点。可见，重投影像点与提取的像点重合的较好。表1给出了4种算法的重投影物方残差、迭代次数和计算时间。重投影物方残差指的是，所有控制点的重投影物方残差的标准差。从重投影物方残差相同可以看出4种算法得到了相同的结果。对比计算时间可以看出本加速算法是有效的。

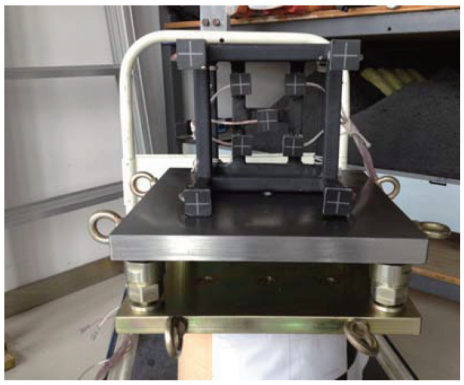

图4　9个控制点结构

Fig. 4　Structure of 9 infrared reference points

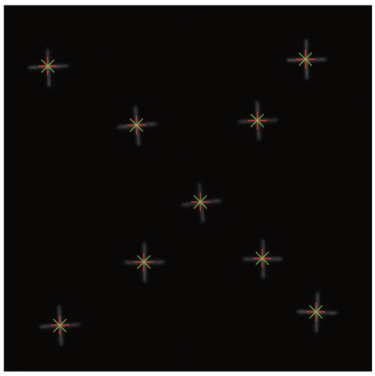

图5 实验图像及提取与重投影结果

Fig.5 Extracted points (red '+') and the reprojection points (green '×') in real image

表1 真实实验结果

Table 1 Results of real data experiments

	Reprojection object-space error /mm	Number of iterations	Computing time /ms
LHM	0.112	12	5.4
LHM+	0.112	12	3.0
RPnP+LHM	0.112	4	4.5
RPnP+LHM+	0.112	4	3.5

4 结 论

针对相机位姿估计的经典正交迭代算法，提出了一种加速正交迭代算法。通过规整正交迭代过程中的运算，使得每一步迭代的计算复杂度从O(n)降低为O(1)，从而大大加快了迭代过程的运算。通过实验，提出了选择RPnP计算初值，再使用加速正交迭代算法进行迭代运算的方法，在控制点不多的情况下，是一种精度接近最大似然估计，且计算速度最快的算法。

由于提出的加速算法的每一步迭代复杂度为常值，因此能够适应更多的迭代，这样，为了解决正交迭代局部最优的问题，可以单独使用多个初值，分别进行迭代，从而更容易找到全局最优解。之后的工作拟改进RPnP，使其可以快速获得多个线性初始解，然后再进行加速正交迭代。

参 考 文 献

1 R Hartley, A Zisserman. Multiple View Geometry in Computer Vision [M]. Cambridge University Press, Second Edition, 2004.

2 J Mcglone, E Mikhail, J Bethel. Manual of Photogrammetry [M]. American Society for Photogrammetry and Remote Sensing, fifth edition, 2004.

3 G N Desouza, A C Kak. Vision for mobil e robot navigation: A survey [J]. IEEE Transations on Pattern Analysis and Machine Intelligence, 2002, 24(2): 237-267.

4 R Azuma, Y Baillot, R Behringer, *et al*.. Recent advances in augmented reality [J]. IEEE Computer Graphics and Applications, 2001, 21(6): 34-47.

5 X S Gao, X R Hou, H F Cheng. Complete solution classification for the perspective-three-point problem [J]. IEEE Transations on Pattern Analysis and Machine Intelligence, 2003, 25(8): 930-943.

6 L Kneip, D Scaramuzza, R Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation [C]. The 24th IEEE Conference on Computer Vision and Pattern Recogoition, 2011. 2969-2976.

7 L Quan, Z Lan. Linear N-point camera pose determination [J]. IEEE Transations on Pattern Analysis and Machine Intelligence, 1999, 21(8): 774-780.

8 P D Fiore. Efficient linear solution of exterior orientation [J]. IEEE Transations on Pattern Analysis and Machine Intelligence, 2001, 23(2): 140-148.

9 A Ansar, K Daniilidis. Linear pose estimation from points or lines [J]. IEEE Transations on Pattern Analysis and Machine Intelligence, 2003, 25(5): 578-589.

10 V Lepetit, F M Noguer, P Fua. EPnP: An accurate O(n) solution to the PnP problem [J]. International Journal of Computer Vision, 2009, 81(2): 155-166.

11 S Q Li, C Xu, M Xie. A robust O(n) solution to the perspective-n-point problem [J]. IEEE Transations on Pattern Analysis and Machine Intelligence, 2012, 34(7): 1444-1450.

12 J A Hesch, S I Roumeliotis. A Direct Least-Squares (DLS) method for PnP [C]. IEEE International Conference on Computer Vision, 2011. 383-390.

13 Y Zheng, Y Kuang, S Sugimoto, *et al*.. Revisiting the PnP problem: A fast, general and optimal solution [C]. IEEE International Conference on Computer Vision, 2013. 2344-2351.

14 D F Demethon, L S Davis. Model-based object pose in 25 lines of code [J]. International Journal of Computer Vision, 1995, 15(1-2): 123-141.

15 C P Lu, G D Hager, E Mjolsness. Fast and globally convergent pose estimation from video images [J]. IEEE Transations on Pattern Analysis and Machine Intelligence, 2000, 22(6): 610-622.

16 Xu Yunxi, Jiang Yunliang, Chen Fang. A generalized orthogonal iterative algorithm for pose estimation of multiple camera systems [J]. Acta Optical Sinica, 2009, 29(1): 72-77.
许允喜, 蒋云良, 陈 方. 多摄像机系统位姿估计的广义正交迭代算法[J]. 光学学报, 2009, 29(1): 72-77.

17 G Schweighofer, A Pinz. Globally optimal O(n) solution to the PnP problem for general camera models [C]. British Machine vision Conference, 2008. 1-10.

18 V Garro, F Crosilla, A Fusiello. Solving the PnP problem with anisotropic orthogonal procrustes analysis [C]. Conference of 3D Imaging, Modeling, Processing, Visccalization and Transmission, 2012. 262-269.

19 R Hartley, F Kahl. Global optimization through rotation space search [J]. International Journal of Computer Vision, 2009, 82(1): 64-79.

20 S Umeyama. Least-squares estimation of transformation parameters between two point patterns [J]. IEEE Transations on Pattern Analysis and Machine Intelligence, 1991, 13(4): 376-380.

21 Z Y Zhang. A flexible new technique for camera calibration [J]. IEEE Transations on Pattern Analysis and Machine Intelligence, 2000, 22(11): 1330-1334

栏目编辑: 张浩佳