Appendices

7. Additional Background Material

7.1. Line Projection Matrices

Let $A \in \mathbb{R}^{3\times 4}$ be a projection matrix, factoring as $KR[I|-t]$ where $R \in SO(3)$ and $t \in \mathbb{R}^3$. Then, the exterior square is

$$\mathcal{P} = \begin{bmatrix} A^2 \wedge A^3 \\ A^3 \wedge A^1 \\ A^1 \wedge A^2 \end{bmatrix}$$

where A^i indicates the ith row of the camera matrix A. Here, $\wedge$ is the wedge product, or the exterior product, between two vectors. The order of the basis is given by the order in the $columnindices$ in the following explanation. This can be calculated specifically in the following way. First, let $rowindices = [2,3;\ 1,3;\ 1,2]$, $columnindices = [1,2;\ 1,3;\ 1,4;\ 2,3;\ 2,4;\ 3,4]$. Then let $\mathcal{P}$ be the 3×6 matrix, where $\mathcal{P}_{i,j}$ is calculated by the minors

$$\det(A(rowindices(i,:), columnindices(j,:))$$

if $i = 1,3$, and by

$$-\det(A(rowindices(i,:), columnindices(j,:)$$

if $i = 2$. When A is a $3n\times 4$ matrix, then let $\mathcal{P}$ denote the $3n\times 6$ matrix of the stacked exterior squares. We refer to [53] for a more detailed description of the exterior algebra of a vector space. Note that the matrix $\mathcal{P}$ is also the line projection matrices, which can project 3D world lines in plucker coordinates (which are in $\mathbb{R}^6$) to lines in the image plane. We refer to [34] for more details on the role of line projection matrices in computer vision.

8. Proofs for Theorems

8.1. Block Trifocal Tensor in Collinear Cases

The reference [27] includes a characterization of the low multilinear rank for the block trifocal tensor under the non-collinear setting. We extend this to the collinear setting, showing that the block trifocal tensor will then have a low multilinear rank of $(5,4,4)$.

Theorem 8.1. *Given a block trifocal tensor with appropriately chosen blockwise scales. If the n cameras that produce $\mathcal{T}^n$ are all collinear and do not all share the same camera center, then mlrank$(\mathcal{T}^n) = (5,4,4)$.*

Proof. Since it has been shown in the proof for Theorem 3.1 that the stacked camera matrices will only exhibit a low rank if all cameras lie at the same point, it is the same case for the low ranks in the flattenings of the second and third mode. We only need to show the rank drop in the first mode. Note that $\mathcal{T}^n_{(1)} = \mathcal{P}(\mathcal{G}_T)_{(1)}(C\otimes C)^T$, we just need to show that $\mathcal{P} \in \mathbb{R}^{3n\times 6}$ has rank 5. Since for individual line projection matrices $\mathcal{P}_i$, we have $\mathcal{P}_i\mathcal{L} = 0$ only when $\mathcal{L}$ is a line that passes through the camera center. Since we have a collinear set of cameras, there is only one line that passes through all of the camera centers, hence the kernel of $\mathcal{P}$ has dimension 1. □

8.2. Proof for Theorem 3.1

Proof. By expanding equation (1) and using the definition of the Tucker product

$$(\mathcal{G}_Q \times_1 C \times_2 C \times_3 C \times_4 C)_{pqrs} = \sum_{a=1}^{4}\sum_{b=1}^{4}\sum_{c=1}^{4}\sum_{d=1}^{4} (\mathcal{G}_Q)_{abcd} C_{ap} C_{bq} C_{cr} C_{ds}, \tag{12}$$

we can see that the core $\mathcal{G}_Q$ admits the following structure:

$$(\mathcal{G}_Q)_{abcd} = \begin{cases} \text{sgn}(abcd) & \text{if } a,b,c,d \in \{1,2,3,4\} \text{ are not all the same} \\ 0 & \text{otherwise.} \end{cases}$$

Then, each quadrifocal tensor block $\mathcal{Q}^n_{ijkl}$ calculated through the Tucker decomposition will be the exact same as the quadrifocal tensor calculated using (1). Now, we establish the full rankness of C in the non-collinear case. Suppose that rank$(C) < 4$, there exists $x \in \mathbb{R}^4$ such that $Cx = 0$. This is equivalent to $P_i x = 0$ for any $i = 1,...,n$. However, $P_i x = 0$ only when x is the camera centre of the camera. This means that x is the camera center for all cameras, which is a contradiction. Thus, if the cameras that produce $\mathcal{Q}^n$ don't share the same camera center, the multilinear rank of $\mathcal{Q}^n = (4,4,4,4)$, even in the collinear case.

□

8.3. Proof for Theorem 3.2

Proof. We prove by explicitly calculating the contractions with two vectors. We know $\mathcal{Q}^n$ admits the factorization $\mathcal{Q}^n = \mathcal{G}_Q \times_1 C \times_2 C \times_3 C \times_4 C$, where $\mathcal{G}_Q \in \mathbb{R}^{4\times4\times4\times4}$. Denote P-Rank($T$) as $(M_1, M_2, M_3, M_4, M_5, M_6)$. Calculating the P-Rank involves taking generic linear combinations of the slices of the tensor, which is equivalent to contracting with a matrix $X \in \mathbb{R}^{3n\times3n}$. For example, for M_1 it can be calculated as the rank for $\mathcal{Q}^n_{p12} = \sum_{i=1}^{3n}\sum_{j=1}^{3n} X_{ij}\mathcal{Q}^n_{ij::}$.

(1) $(\mathbf{M_1})$ Recall that

$$\mathcal{Q}^n_{ijkl} = (\mathcal{G}_Q \times_1 C \times_2 C \times_3 C \times_4 C)_{ijkl} = \sum_{a=1}^{4}\sum_{b=1}^{4}\sum_{c=1}^{4}\sum_{d=1}^{4}\mathcal{G}_{abcd}C_{ia}C_{jb}C_{kc}C_{ld}.$$

Then

$$(\sum_{i,j} X_{ij}\mathcal{Q}^n_{ij::})_{kl} = \sum_{i,j} X_{ij}\sum_{a=1}^{4}\sum_{b=1}^{4}\sum_{c=1}^{4}\sum_{d=1}^{4}\mathcal{G}_{abcd}C_{ia}C_{jb}C_{kc}C_{ld} = \sum_{a=1}^{4}\sum_{b=1}^{4}\sum_{c=1}^{4}\sum_{d=1}^{4}\mathcal{G}_{abcd}(\sum_{i,j} X_{ij}C_{ia}C_{jb})C_{kc}C_{ld}.$$

Let $S_{ab} = \sum_{i,j} X_{ij}C_{ia}C_{jb}$, or in other words $S = C^T XC$. Then, we have

$$\begin{aligned}(\sum_{i,j} X_{ij}\mathcal{Q}^n_{ij::})_{kl} &= \sum_{a=1}^{4}\sum_{b=1}^{4}\sum_{c=1}^{4}\sum_{d=1}^{4}\mathcal{G}_{abcd}S_{ab}C_{kc}C_{ld} \\ &= \sum_{c=1}^{4}\sum_{d=1}^{4}\left(\sum_{a=1}^{4}\sum_{b=1}^{4}\mathcal{G}_{abcd}S_{ab}\right)C_{kc}C_{ld} = \left(C\left(\sum_{a=1}^{4}\sum_{b=1}^{4}\mathcal{G}_{abcd}S_{ab}\right)C^T\right)_{kl}.\end{aligned}$$

Here, $\mathcal{G}_S \in \mathbb{R}^{4\times4}$ is given by

$$\mathcal{G}_S = \sum_{a=1}^{4}\sum_{b=1}^{4}\mathcal{G}_{abcd}S_{ab} = \begin{bmatrix} 0 & S_{34}-S_{43} & S_{42}-S_{24} & S_{23}-S_{32} \\ S_{43}-S_{34} & 0 & S_{14}-S_{41} & S_{31}-S_{13} \\ S_{24}-S_{42} & S_{41}-S_{14} & 0 & S_{12}-S_{21} \\ S_{32}-S_{23} & S_{13}-S_{31} & S_{21}-S_{12} & 0 \end{bmatrix}$$

By our definition of projection rank, rank$(X) = 1$, so $X = xy^T$ for some generic $x, y \in \mathbb{R}^4$. In general, we can explicitly calculate the rank of $\mathcal{G}_S$, where the reduced row echelon form is

$$\text{rref}(\mathcal{G}_S) = \begin{bmatrix} 1 & 0 & -(x_1y_4-x_4y_1)/(x_3y_4-x_4y_3) & -(x_1y_3-x_3y_1)/(x_3y_4-x_4y_3) \\ 0 & 1 & -(x_2y_4-x_4y_2)/(x_3y_4-x_4y_3) & -(x_2y_3-x_3y_2)/(x_3y_4-x_4y_3) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

when $x_3y_4 - x_4y_3 \neq 0$. Thus, for generic X, rank$(\mathcal{G}_S) = 2$, so that rank$(C\mathcal{G}_S C^T) = 2$, proving the projection rank for M_1 to be 2. This has also been checked numerically and symbolically.

(2) $\mathbf{M_2}, \mathbf{M_3}, \mathbf{M_4}, \mathbf{M_5}, \mathbf{M_6}$: Due to symmetries, M_2,...,M_6 will have the same ranks and properties. We omit the details.

This establishes Theorem 3.2. □

8.4. Proof for Proposition 3.3

Proof. These are mainly facts from [34], but we still describe the rationale and how it corresponds to the descriptions in the reference. Recall that the quadrifocal tensor can be calculated via the following formula (1) given four cameras P_i,P_j,P_k,P_l:

$$(\mathcal{Q}^n_{ijkl})_{pqrs} = \det\begin{bmatrix} P_i^p \\ P_j^q \\ P_k^r \\ P_l^s \end{bmatrix}.$$

(1) For $\mathcal{Q}^n_{iiii}$ diagonal blocks, all four cameras will be the same in the definition for quadrifocal tensors from camera matrices. Since P_i only has 3 rows, there must be a repeating row in the determinant, meaning that $\mathcal{Q}^n_{iiii} = 0$ in $\mathbb{R}^{3\times3\times3\times3}$.

(2) Suppose that we now have 3 overlapping indices, such that we are considering the blocks $\mathcal{Q}^n_{iiij},\mathcal{Q}^n_{iiji},\mathcal{Q}^n_{ijii},\mathcal{Q}^n_{jiii}$ for $i\neq j$. Then, among the four rows in the determinant, three must correspond to the same camera i. By [34], these entries will be the epipoles up to signs, where epipoles are the images of the camera center of view i in view j.

(3) Suppose that we now have 2 overlapping indices, such that we are looking at the blocks $\mathcal{Q}^n_{iijk}$, $\mathcal{Q}^n_{ijik}$, $\mathcal{Q}^n_{ijki}$, $\mathcal{Q}^n_{jiik}$, $\mathcal{Q}^n_{jiki}$, $\mathcal{Q}^n_{jkii}$, $\mathcal{Q}^n_{iikj}$, $\mathcal{Q}^n_{ikij}$, $\mathcal{Q}^n_{ikji}$, $\mathcal{Q}^n_{kiij}$, $\mathcal{Q}^n_{kiji}$, $\mathcal{Q}^n_{kjii}$ for $i\neq j,i\neq k,j\neq k$. Then, among the four rows in the determinant, two correspond to the same camera i. Again by [34], these entries will be the elements of the trifocal tensor up to signs.

(4) Suppose that we now have 2 overlapping indices, and the other two are also the same, such that we are looking at the blocks, $\mathcal{Q}^n_{iijj},\mathcal{Q}^n_{ijij},\mathcal{Q}^n_{ijji},\mathcal{Q}^n_{jiij},\mathcal{Q}^n_{jiji},\mathcal{Q}^n_{jjii}$ for $i\neq j$. Then, among the four rows in the determinant, two correspond to the same camera i, and the other two correspond to the same camera j. Again by [34], these entries will be the elements of the fundamental matrices up to signs.

This completes the proof. □

8.5. Proof for Theorem 3.4

Proof. Blockwise multiplication by a rank-1 tensor with non-vanishing entries preserves multilinear rank, as this operation is equivalent to a Tucker product with invertible diagonal matrices.

Hence, without loss of generality, assume $\lambda_{i111}=\lambda_{1j11}=\lambda_{11k1}=\lambda_{111\ell}=1$ for all $i,j,k,\ell\in\{2,\dots,n\}$. We will show below that for some $c\in\mathbb{R}^*$, the entries satisfy:

- $\lambda_{ijk\ell}=c$ if exactly two of i,j,k,ℓ equal 1;
- $\lambda_{ijk\ell}=c^2$ if exactly one of i,j,k equals 1;
- $\lambda_{ijk\ell}=c^3$ if none of i,j,k,ℓ equal 1 and the indices are not all the same.

This result will establish the theorem, since setting $\alpha=\beta=\gamma=(1,c,\dots,c)$ and $\delta=(\frac{1}{c},1,\dots,1)$ ensures $\lambda_{ijk\ell}=\alpha_i\beta_j\gamma_k\delta_\ell$ whenever i,j,k,ℓ are not all equal.

We let $\mathcal{Q}^n_{(1)}$ and $(\lambda\odot_b\mathcal{Q}^n)_{(1)}$ denote the mode-1 matrix flattenings in $\mathbb{R}^{3n\times 27n^3}$ of the block quadrifocal tensor and its scaled variant, where rows correspond to the first mode of the tensors. Invoking Theorem 3.1 alongside our standing assumptions shows that both matrices have rank 4. Consequently, every 5×5 minor of these matrices must be zero.

We will exploit this by analyzing specific 5×5 submatrices of $(\lambda\odot_b\mathcal{Q}^n)_{(1)}$ to construct a system of constraints on λ, ultimately establishing the existence of the constant c. We adopt the index convention (ip) for rows and $(jq,kr,\ell s)$ for columns, where $i,j,k,\ell\in[n]$ and $p,q,r,s\in[3]$. Thus, entries satisfy $((\lambda\odot_b\mathcal{Q}^n)_{(1)})_{(ip),(jq,kr,\ell s)}=\lambda_{ijk\ell}(\mathcal{Q}^n_{ijk\ell})_{pqrs}$.

Case 1: The first submatrix of $(\lambda\odot_b\mathcal{Q}^n)_{(1)}$ we consider has column indices $(i1,13,12)$, $(12,j2,11)$, $(12,j3,11)$, $(13,j3,12)$, $(11,j1,13)$ and row indices (11), (12), (13), $(i1)$, $(i2)$, where $i,j\in\{2,\dots,n\}$. Explicitly, the submatrix is

$$\begin{bmatrix}
(\mathcal{Q}^n_{1i11})_{1132} & (\mathcal{Q}^n_{11j1})_{1221} & (\mathcal{Q}^n_{11j1})_{1231} & (\mathcal{Q}^n_{11j1})_{1332} & (\mathcal{Q}^n_{11j1})_{1113}\\
(\mathcal{Q}^n_{1i11})_{2132} & (\mathcal{Q}^n_{11j1})_{2221} & (\mathcal{Q}^n_{11j1})_{2231} & (\mathcal{Q}^n_{11j1})_{2332} & (\mathcal{Q}^n_{11j1})_{2113}\\
(\mathcal{Q}^n_{1i11})_{3132} & (\mathcal{Q}^n_{11j1})_{3221} & (\mathcal{Q}^n_{11j1})_{3231} & (\mathcal{Q}^n_{11j1})_{3332} & (\mathcal{Q}^n_{11j1})_{3113}\\
\lambda_{ii11}(\mathcal{Q}^n_{ii11})_{1132} & \lambda_{i1j1}(\mathcal{Q}^n_{i1j1})_{1221} & \lambda_{i1j1}(\mathcal{Q}^n_{i1j1})_{1231} & \lambda_{i1j1}(\mathcal{Q}^n_{i1j1})_{1332} & \lambda_{i1j1}(\mathcal{Q}^n_{i1j1})_{1113}\\
\lambda_{ii11}(\mathcal{Q}^n_{ii11})_{2132} & \lambda_{i1j1}(\mathcal{Q}^n_{i1j1})_{2221} & \lambda_{i1j1}(\mathcal{Q}^n_{i1j1})_{2231} & \lambda_{i1j1}(\mathcal{Q}^n_{i1j1})_{2332} & \lambda_{i1j1}(\mathcal{Q}^n_{i1j1})_{2113}
\end{bmatrix},$$

which we abbreviate as

$$\begin{bmatrix}
* & * & * & * & *\\
* & * & * & * & *\\
* & * & * & * & *\\
\lambda_{ii11}* & \lambda_{i1j1}* & \lambda_{i1j1}* & \lambda_{i1j1}* & \lambda_{i1j1}*\\
\lambda_{ii11}* & \lambda_{i1j1}* & \lambda_{i1j1}* & \lambda_{i1j1}* & \lambda_{i1j1}*
\end{bmatrix}, \tag{13}$$

with asterisk denoting the corresponding entry in $\mathcal{Q}^n_{(1)}$. We view the determinant of (13) as a polynomial with respect to λ. It has degree ≤ 2 in the variables $\lambda_{ii11},\lambda_{i1j1}$. Observe that if $\lambda_{i1j1}=0$, the bottom two rows of the matrix are linearly independent. Also if $\lambda_{i1j1}-\lambda_{ii11}=0$, then (13) equals a 5×5 submatrix of $\mathcal{Q}^n_{(1)}$ with rows operations performed; therefore (13) is rank-deficient. It follows that the determinant of (13) takes the form

$$s\lambda_{i1j1}(\lambda_{i1j1}-\lambda_{ii11}).$$

Here the scale $s=s(P_1,P_i,P_j)$ is a polynomial in the camera matrices. Due to polynomiality, s is nonzero Zariski-generically if we can exhibit a *single* instance of matrices P_1,P_i,P_j where the determinant of (13) does not vanish identically for all $\lambda_{i1j1},\lambda_{ii11}$.

Furthermore, we just need an instance with $i=j$, as this corresponds to a specialization of the case $i\neq j$. Computational verification with a random numerical instance of P_1,P_i proves the non-vanishing. Recalling the standing assumptions, we deduce $\lambda_{i1j1}=\lambda_{ii11}$.

We apply the same argument to modewise permutations of $\lambda\odot_b\mathcal{Q}^n$ and $\mathcal{Q}^n$, and obtain

$$\lambda_{\pi(i1j1)}=\lambda_{\pi(ii11)} \quad \text{for all } i,j\in\{2,\ldots,n\} \text{ and permutations } \pi.$$

The argument goes through as $\pi\cdot\mathcal{Q}^n$ and $\pi\cdot(\lambda\odot_b\mathcal{Q}^n)$ have multilinear ranks bounded by $(4,4,4,4)$ and from equation 4. So (13) looks the same but with indices permuted and possibly a sign flip.

We now see that λ-entries with two 1-indices agree. Indeed, taking $i=j$ above gives $\lambda_{\pi_1(i1i1)}=\lambda_{\pi_2(ii11)}$ for all π_1 and π_2 that fix $(ii11)$ and $(i1i1)$ respectively. So $\lambda_{ii11}=\lambda_{\pi(ii11)}$ for all π. Taking $i\neq j$ gives $\lambda_{ii11}=\lambda_{\pi(i1j1)}=\lambda_{jj11}$ for all π. Together, there exists $c\in\mathbb{R}^*$ such that $c=\lambda_{\pi(ij11)}$ for all $i,j\in\{2,\ldots,n\}$ and permutations π.

Case 2: Next we consider the submatrix of $(\lambda\odot_b\mathcal{Q}^n)_{(1)}$ with column indices $(j1,k3,12)$, $(12,j2,11)$, $(12,j3,11)$, $(13,j3,12)$, $(11,j1,13)$ and row indices (11), (12), (13), $(i1)$, $(i2)$, where $i,j,k\in\{2,\ldots,n\}$. It looks like

$$\begin{bmatrix} c* & * & * & * & * \\ c* & * & * & * & * \\ c* & * & * & * & * \\ \lambda_{ijk1}* & c* & c* & c* & c* \\ \lambda_{ijk1}* & c* & c* & c* & c* \end{bmatrix}, \tag{14}$$

where asterisks denote corresponding entries in $\mathcal{Q}^n_{(1)}$. As a polynomial in c and λ_{ijk1}, the determinant of (14) is a scalar multiple of $c(c^2-\lambda_{ijk1})$. This is because the polynomial has degree ≤ 3, if $c=0$ then the bottom two rows of (14) are linearly dependent, and if $c^2=\lambda_{ijk1}$ then (14) is a 5×5 submatrix of $\mathcal{Q}^n_{(1)}$ with row and column operations performed. The scale is a polynomial in P_1,P_i,P_j,P_k. It is Zariski-generically nonzero if we exhibit one instance of camera matrices such that the determinant of (13) does not vanish for all c,λ_{ijk1}. Further, it suffices to find an instance where $i=j=k$, as all other cases specialize to this. Computational verification with a random numerical instance of P_1,P_i proves the non-vanishing. It follows that $c^2=\lambda_{ijk1}$. Appealing to symmetry like before, $c^2=\lambda_{\pi(ijk1)}$ for all $i,j,k\in\{2,\ldots,n\}$ and permutations π. Summarizing, all λ-entries with a single 1-index equal c^2.

Case 3: Consider the submatrix of $(\lambda\odot\mathcal{Q}^n)_{(1)}$ with columns $(j1,k3,\ell 2)$, $(12,i2,11)$, $(12,i3,11)$, $(13,i3,12)$, $(11,i1,13)$ and rows (11), (12), (13), $(i1)$, $(i2)$, where $i,j,k,\ell\in\{2,\ldots,n\}$ and i,ℓ are distinct. The submatrix looks like

$$\begin{bmatrix} c^2* & * & * & * & * \\ c^2* & * & * & * & * \\ c^2* & * & * & * & * \\ \lambda_{ijk\ell}* & c* & c* & c* & c* \\ \lambda_{ijk\ell}* & c* & c* & c* & c* \end{bmatrix}. \tag{15}$$

The determinant of (15) is $c(c^3-\lambda_{ijk\ell})$ multiplied by a polynomial in P_1,P_i,P_j,P_k,P_ℓ. The most specialized case is $i=j=k$. Computer verification with a random numerical instance proves the polynomial is not identically zero. We deduce that $c^3=\lambda_{ijk\ell}$. By symmetry, $c^3=\lambda_{\pi(ijk\ell)}$ for all $i,j,k,\ell\in\{2,\ldots,n\}$ with i,ℓ distinct and all permutations π. In other words, λ-entries with no 1-indices and non-identical indices equal c^3.

Putting cases 1, 2 and 3 together shows that λ takes the announced form. This proves Theorem 3.4. □

8.6. Proof for Theorem 4.1

Proof. The proof of this theorem follows from explicit calculation. We just need to verify that

$$\mathcal{E}^n_{ij}=\mathcal{P}_i\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}\mathcal{P}_j^T.$$

For each entry $(\mathcal{E}_{ij}^n)_{kl}$ in $\mathcal{E}_{ij}^n$,

$$(\mathcal{E}_{ij}^n)_{kl} = (-1)^{k+l}\det\begin{bmatrix}\sim P_i^l \\ \sim P_j^k\end{bmatrix} = (\mathcal{P}_i)_k \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} (\mathcal{P}_j)_l^T$$

by expanding every entry. A direct expansion shows the result. This has also been checked numerically. □

8.7. Proof for Theorem 4.2

Proof. We prove this by explicit calculation. The block tensor $\mathcal{T}^n$ admits the factorization $\mathcal{T}^n = \mathcal{G} \times_1 \mathcal{P} \times_2 C \times_3 C$, where $\mathcal{G} \in \mathbb{R}^{6\times4\times4}$. Denote P-Rank($\mathcal{T}^n$) as (M_1, M_2, M_3). Calculating the P-Rank involves taking genericlinear combinations of the slices of the tensor, this is equivalent to contracting with a vector. In the tensor-matrix product, let $x \in \mathbb{R}^{3n\times1}$, $\sum_{i=1}^{3n} x_i \mathcal{T}_{ijk}^n = \mathcal{G} \times_1 (x^T P) \times_2 C \times_3 C$. Then we explicitly calculate M_1 from $\mathcal{G} \times_1 (x^T P)$. The same process holds for modes 2 and 3 to compute M_2 and M_3 in the P-Rank.

(1) $\mathbf{M_1}$: Let $y = x^T P$. We have

$$\mathcal{T}_1^n = \sum_{i=1}^{4} y_i \mathcal{G}_{ijk} = \begin{bmatrix} 0 & y_6 & -y_5 & y_4 \\ -y_6 & 0 & y_3 & -y_2 \\ -y_5 & -y_3 & 0 & y_1 \\ -y_4 & y_2 & -y_1 & 0 \end{bmatrix}.$$

Generically, this matrix $\mathcal{T}_1^n$ will clearly have rank 4.

(2) $\mathbf{M_2}$: Let $y = x^T C$, then we have

$$\mathcal{T}_2^n = \sum_{j=1}^{4} y_j \mathcal{G}_{ijk} = \begin{bmatrix} 0 & 0 & 0 & y_4 & -y_3 & y_2 \\ 0 & -y_4 & y_3 & 0 & 0 & -y_1 \\ y_4 & 0 & -y_2 & 0 & y_1 & 0 \\ -y_3 & y_2 & 0 & -y_1 & 0 & 0 \end{bmatrix}$$

This matrix generically will clearly have rank 3. Denote R_i as the i-th row of $\mathcal{T}_2^n$. We can observe that the fourth row is a linear combination of rows 1,2,3, where $y_4 R_4 = y_1 R_1 + y_2 R_2 + y_3 R_3$. However, R_1, R_2, R_3 must be linearly independent in general.

(3) $\mathbf{M_3}$: This is the same as M_2 due to symmetry.

□

9. Algorithm Details

9.1. QuadSync

We present pseudocode for QuadSync in Algorithm 1 in this section.

We count the storage and flop complexities of QuadSync. Let n be the number of cameras. Then the input of QuadSync consists of a block quadrifocal tensor with up to $81n^4$ elements. The auxiliary variables consist of C_1, C_2, C_3, C_4, Λ, W, B, Γ_1, Γ_2, Γ_3, with size $12n\times8+2|\Omega| = O(n^4)$. Thus, the overall storage complexity is $O(n^4)$.

For the flop complexity, we first account for the initialization. For the initialization, C is calculated via HOSVD of a matrix of size $3n \times 27n^3$ where the target rank is 4. This costs costs $O(4\times81\times n^4) = O(n^4)$ flops. For the running iterations, to calculate C, one has to multiply a matrix of size $4\times27n^3$ with a diagonal matrix of size $27n^3 \times 27n^3$, then take the product and multiply it with another matrix of size $27n^3 \times 4$. This multiplication of three matrix will have to be done for a total of n times for each update of the stacked camera matrices C. Thus, the flop complexity for calculating the updates for C is $O(n^4)$. Calculating Λ_Q, W also requires $O(|\Omega|) = O(n^4)$ flops. Calculating B, Γ_i requires $O(n)$ flops. The overall flop complexity is $O(n^4)$.

9.2. Joint Opt.

Though similar to the above, we include detailed calculations for the steps of Joint Opt. and include pseudocode in Algorithm 2. We first give explicit formulas for all the updates that we use.

Algorithm 1 Quadsync IRLS-ADMM

Input: $\mathcal{Q}^n \in \mathbb{R}^{3n\times 3n\times 3n\times 3n}$, $\rho > 0 \in \mathbb{R}$
Ω set of observed block indices
Output: $\bar{C} \in \mathbb{R}^{3n\times 4}$
Normalize $\mathcal{Q}^n$ so that each block has norm 1
Obtain C_i, B from first four singular vectors in first factor matrix of HOSVD($\mathcal{Q}^n$)
Calculate initial IRLS weights W_Q from (5)
while not converged **do**
  # Now we run the ADMM cases
  while not converged **do**
    # First optimize for C_i,Λ alternatingly
    while not converged **do**
      Update C_i, $i=1,2,3,4$ via (6)
      Update Λ_Q via (7)
    end while
    Update B via (8)
    Update Γ_i, $i=1,2,3,4$ via (9)
  end while
  Update new IRLS weights W_Q via (5)
end while
$\bar{C} = \text{avg}(C_1, C_2, C_3, C_4)$
Return

(1) $\mathbf{W_Q}$, $\mathbf{W_T}$, $\mathbf{W_E}$: Here W is the set of IRLS weights integrated in the norm, so

$$w_{ijkl} = \begin{cases} 1/m^t_{ijkl} & \text{if } (i,j,k,l) \in \Omega \\ 0 & \text{otherwise.} \end{cases}$$

For W_Q we have

$$(m_Q)^t_{ijkl} = \max(\delta, sqrt(\|(\Lambda_Q)^{(t-1)}_{ijkl}\tilde{\mathcal{Q}}^n_{ijkl} - \mathcal{G}_Q \times_1 (C_1)^{(t-1)}_i \times_2 (C_2)^{(t-1)}_j \times_3 (C_3)^{(t-1)}_k \times_4 (C_4)^{(t-1)}_l)\|_F). \tag{16}$$

For W_T we have

$$(m_T)^t_{ijkl} = \max(\delta, sqrt(\|(\Lambda_T^{(t-1)})_{ijkl}\tilde{\mathcal{T}}^n_{ijkl} - \mathcal{G}_T \times_1 (P_1)^{(t-1)}_i \times_2 (C_5)^{(t-1)}_j \times_3 (C_6)^{(t-1)}_k\|_F). \tag{17}$$

For W_E we have

$$(m_E)^t_{ijkl} = \max(\delta, sqrt(\|(\Lambda_E)^{(t-1)}_{ijkl}\tilde{\mathcal{E}}^n_{ijkl} - \mathcal{G}_E \times_1 (P_2)^{(t-1)}_i \times_2 (P_3)^{(t-1)}_j\|_F). \tag{18}$$

(2) $\mathbf{C_1}$, $\mathbf{C_2}$, $\mathbf{C_3}$, $\mathbf{C_4}$: For the ith mode, let $C_{\sim i} = C_4 \otimes \cdots \otimes C_{i+1} \otimes C_{i-1} \otimes \cdots \otimes C_1$, and let $K = (\mathcal{G}_Q)_{(i)} C^T_{\sim i}$. Then, let x_j denote the ith row of the variable being optimized for. Then

$$x_j = (\frac{\rho}{2}(B - \Gamma_i)_j + (\frac{1}{n_Q}(W^2_{(i)})_j \odot_b [(\Lambda \odot_b \tilde{\mathcal{Q}}^n)_{(i)}]_j)K^T)(\frac{\rho}{2} I_{4\times 4} + \frac{1}{n_Q} K \text{diag}((W^2_{(i)})_j) K^T)^{-1}. \tag{19}$$

(3) $\mathbf{C_5}$, $\mathbf{C_6}$, $\mathbf{P_1}$: To solve for C_i, $i=5,6$, let $C_{\sim i} = P \otimes C_j$, and let $K = (\mathcal{G}_T)_{(i)} C^T_{\sim i}$. Then, we have the update rule for C_i as

$$x_j = (\frac{\rho}{2}(B - \Gamma_i)_j + \frac{1}{n_T}((W^2_T \odot L_T) \odot_b T)_j K^T)(\frac{\rho}{2} I_{4\times 4} + \frac{1}{n_T} K \text{diag}((W_T)^2_j) K^T)^{-1}. \tag{20}$$

For P_1, let $K = (\mathcal{G}_T)_{(3)}(C_2 \otimes C_1)^T$. Then

$$x_j = (\frac{\rho}{2}(D - \tau_i)_j + \frac{1}{n_T}((W^2_T \odot L_T) \odot_b T)_j K^T)(\frac{\rho}{2} I_{6\times 6} + \frac{1}{n_T} K \text{diag}(W^2_T) K^T)^{-1}. \tag{21}$$

(4) $\mathbf{P_2}$, $\mathbf{P_3}$: To solve for P_i, $i=2,3$, let $K=(\mathcal{G}_E)_{(i)}P_j^T$ where $j\neq i$. Then, we have the update rule for P_i as

$$x_j=(\frac{\rho}{2}(D-\tau_i)_j+\frac{1}{n_T}((W_E^2\odot L_E)\odot_b E)_j K^T)(\frac{\rho}{2}I_{6\times 6}+\frac{1}{n_E}K\text{diag}(W_E^2)K^T)^{-1}. \tag{22}$$

(5) $\mathbf{\Lambda_Q}$, $\mathbf{\Lambda_T}$, $\mathbf{\Lambda_E}$: For Λ, we can solve as usual by directly solving the convex optimization problem associated with f_Q, g_T, h_E, This is done by solving for each block separately, where

$$(\Lambda_Q)_{ijkl}=\frac{trace((\llbracket \mathcal{G}_Q; C_1, C_2, C_3, C_4\rrbracket_{ijkl})_{(1)}^T((\tilde{\mathcal{Q}}^n)_{ijkl}))_{(1)})}{\|((\tilde{\mathcal{Q}}^n)_{ijkl})_{(1)}\|_F^2}, \tag{23}$$

$$(\Lambda_T)_{ijk}=\frac{trace((\llbracket \mathcal{G}_T; P_1, C_5, C_6\rrbracket_{ijk})_{(1)}^T((\tilde{\mathcal{T}}^n)_{ijk}))_{(1)})}{\|((\tilde{\mathcal{T}}^n)_{ijk})_{(1)}\|_F^2}, \tag{24}$$

$$(\Lambda_E)_{ij}=\frac{trace((\llbracket \mathcal{G}_E; P_2, P_3\rrbracket_{ij})_{(1)}^T((\tilde{\mathcal{E}}^n)_{ij}))_{(1)})}{\|((\tilde{\mathcal{E}}^n)_{ij})_{(1)}\|_F^2}. \tag{25}$$

After calculating Λ_Q, Λ_T, Λ_E, for each of them, we first symmetrize, then normalize Λ so that $\|\Lambda\|_F^2=1$ and Λ satisfies the required symmetry requirements.

(6) $\mathbf{B}$,$\mathbf{D}$: For B, we can solve directly, where

$$B=\frac{1}{6}\left(\sum_{i=1}^{6}C_i+\Gamma_i\right). \tag{26}$$

For D, we also solve directly, where

$$D=\frac{1}{3}\left(\sum_{i=1}^{3}P_i+\tau_i\right). \tag{27}$$

(7) $\mathbf{\Gamma_i}$, $\tau_\mathbf{i}$: For the final ascent cases for Γ_i, τ_i, we set

$$\Gamma_i=\Gamma_i+(C_i-B), \tag{28}$$

$$\tau_i=\tau_i^{(k)}+(P_i-D). \tag{29}$$

We note that in the joint optimization, we only constrain the equality between C_i's and P_i's. However, there is also a relationship between C_i and P_i, where P_i's elements should be the 2×2 minors of C_i. In future work, this constraint could be further explored to potentially improve the algorithm and the linkage between the factor matrices and the three block entities.

10. Additional Experiments

10.1. Randomized Updates in QuadSync

We test the effect of randomized updates for C_i in QuadSync for the ETH3D 'relief' dataset. Our 'relief' dataset contains 13 images, and the maximum number of columns for updating C_i is $27\times 13^3=59319$. We try randomized updates using columns whose number range from 20 to 59319. Using 30 random columns achieve similar accuracy with notable speed-ups. See Figure 4 for the effect of the number of columns on accuracy and runtime.

10.2. Collinear Experiments

Given a set of images, it is well known that the locations can only be uniquely determined up to a global transformation, only when the graph is *parallel rigid*, see [54] and [55] for more details on parallel rigidity in computer vision. The most degenerate case is when all of the cameras are collinear, meaning that they lie on a common line. In this case, synchronization algorithms that involve solving the location synchronization problem using solely pairwise directions, will fail. However, collinear or close to collinear situations are common in real life, such as in self-driving cars or robot motion. We conduct a small set of synthetic experiments to demonstrate that our algorithm can successfully recover camera poses in the collinear setting. A simple diagram can show the situation, see Figure 5.

Algorithm 2 Joint Opt. IRLS-ADMM

Input: $\mathcal{Q}^n \in \mathbb{R}^{3n\times 3n\times 3n\times 3n}$, $\mathcal{T}^n \in \mathbb{R}^{3n\times 3n\times 3n}$, $\rho > 0$ in $\mathbb{R}$
$\mathcal{E}^n \in \mathbb{R}^{3n\times 3n\times 3n}$, Ω observed indices
Output: $\bar{C} \in \mathbb{R}^{3n\times 4}$
Normalize $\mathcal{Q}^n$, $\mathcal{T}^n$, $\mathcal{E}^n$ so that each block has norm 1
Calculate n_Q, n_T, n_E as the number of estimated blocks respectively.
Obtain C_i, B from first four singular vectors in first factor matrix of HOSVD($\mathcal{Q}^n$)
Calculate initial IRLS weights W_Q, W_T, W_Q from (16), (17), (18).
while not converged **do**
  # Now we run the ADMM steps
  while not converged **do**
    # First optimize for C_i, P_i, Λ_Q, Λ_T, Λ_E alternatingly
    while not converged **do**
      Update all C_i, P_j, for $j = 1,2,3$ via (19), (20), (21), (22)
      Update Λ_Q, Λ_T, Λ_E via (23), (24), (25)
    end while
    Update B via (26)
    Update D via (27)
    Update Γ_i via (28)
    Update τ_i via (29)
  end while
  Update new IRLS weights W_Q, W_T, W_E via (16), (17), (18)
end while
$\bar{C} = \text{avg}(C_1, C_2, C_3, C_4)$
Return

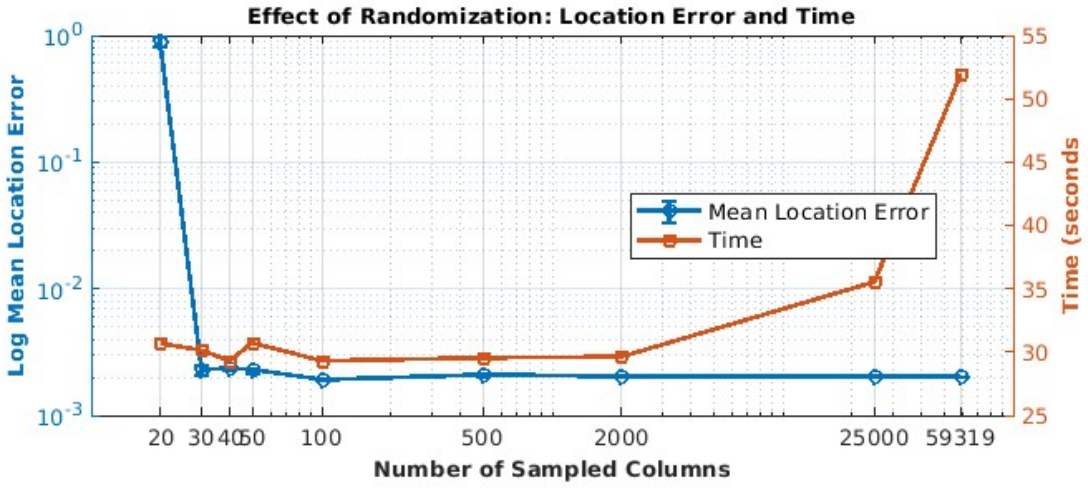

Figure 4. Randomized updates in QuadSync tested on ETH3D 'relief' dataset.

We randomly generate calibrated collinear cameras. Then, for each individual quadrifocal tensor estimation, we add random noise of a certain percentage to its corresponding cameras and form the corresponding block quadrifocal tensor individually for each quadruplet. The noise parameter corresponds to the amount of noise we add to the cameras. We then randomly sample a certain percentage of quadruplets. We discard all the trifocal tensor and fundamental matrices. The results are then aggregated in Table 1. Our algorithm can still recover the camera parameters effectively in the collinear setting. Standard pipelines that synchronization rotation and location separately will fail in this situation. Global synchronization algorithms using the block essential matrix and the block trifocal tensors will need special tuning, as the ranks of the entities drop. The block quadrifocal tensor is insensitive to

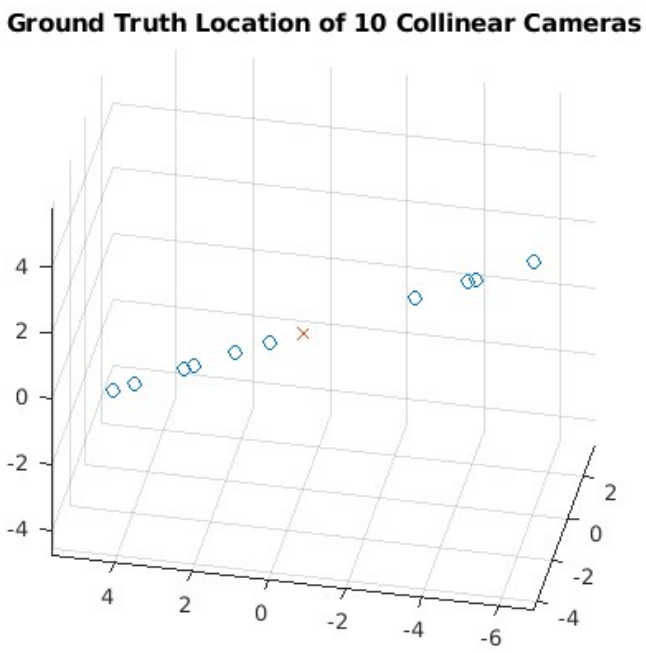

Figure 5. Ground truth location of 10 cameras. × is the origin. ○ denotes the camera centers.

perc. (%)	noise(%)	$\bar{e_t}$	$\hat{e_t}$	$\bar{e_r}$	$\hat{e_r}$
100	0.00	0.00	0.00	0.00	0.00
100	1	0.04	0.04	0.23	0.18
100	5	0.24	0.22	2.67	2.52
80	0	0.00	0.00	0.00	0.00
80	1	0.04	0.03	0.37	0.37
80	5	0.36	0.36	3.07	2.68
60	0	0.00	0.00	0.00	0.00
60	1	0.06	0.05	0.42	0.38
60	5	0.64	0.54	4.52	3.67

Table 1. Results for synthetic experiments with collinear cameras. $\bar{e_t}$: mean location error. $\hat{e_t}$: median location error. $\bar{e_r}$: mean rotation error. $\hat{e_r}$: median rotation error.

such cases and has a special advantage in the collinear motion setting.

10.3. Distributed Synchronization

Our quadrifocal tensor synchronization algorithm becomes slow as it deals with higher order tensors. However, it can be sped up with distributed synchronization, where clusters are formed and solved in parallel. In our set of experiments, we follow the way we add noise to the cameras in the collinear experiments. We artificially handpick the clusters, so that each cluster is fully dense (we continue to disregard trifocal and fundamental matrix elements). There are overlapping cameras in each cluster, and we finally synchronize all the clusters by calculating the projective transformation that maps the overlapping cameras to be in the same projective frame. The main take away for this experiment is to demonstrate the potential for applicability of the algorithm on large datasets. We conduct sets of experiments with just 2 CPU cores, and show that by focusing on smaller subsets of the data and using parallelization, we can greatly reduce the runtime of our algorithm, making it more scalable and applicable to larger scenes.

Experiment: We compare just the difference in runtime for a small synthetic dataset. We add no noise and assume that the entire block is observed accurately. We break the entire set of 30 cameras into 3 clusters. The second and first cluster have 5 overlapping

cameras, same with the second and third. We then synchronize each cluster separately. The resulting cameras will be in different projective frames. We then use the overlapping cameras to align the clusters to the same projective frame. Noise is added similarly as in the synthetic collinear experiments. We include the results in Table 2.

We also include a small experiment with handpicked clusters for the EPFL CastleP30 dataset. Note that we can not directly synchronization CastleP30 using QuadSync, as the cameras point outwards around a courtyard, so that each camera only sees a small portion of the courtyard, and the higher-order viewing graph is very sparse. However, applying QuadSync on each of the handpick clusters then merging the results produces great reconstruction for the CastleP30 dataset. See Figure 6 for a comparison of the retrieved poses and ground truth poses.

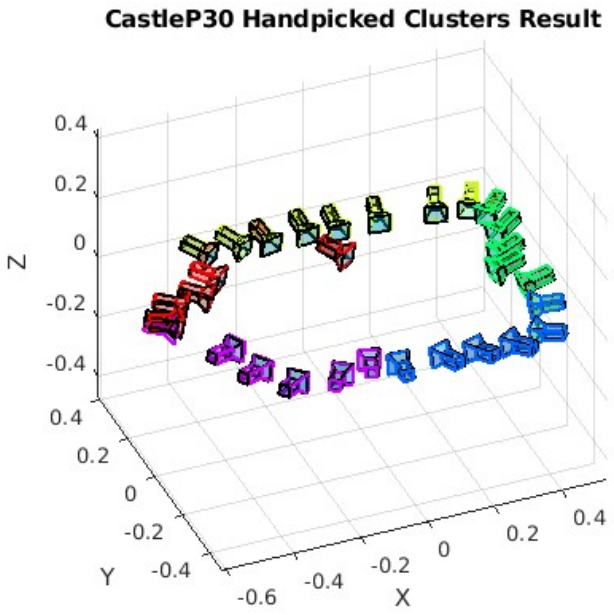

Figure 6. Retrieved poses (colored, where each cluster has a distinct color) vs. ground truth poses (black) for CastleP30 with handpicked clusters.

size	noise	$\bar{e_t}$	$\hat{e_t}$	$\bar{e_r}$	$\hat{e_r}$	time(s)
Full (30 cams)	0	0	0	0	0	1666.3
Cluster1 (10 cams)	0	0	0	0	0	27.3
Cluster2 (15 cams)	0	0	0	0	0	92.6
Cluster3 (15 cams)	0	0	0	0	0	97.7
Aligned (30 cams)	0	0	0	0	0	150.2
Full (30 cams)	1	0.11	0.10	0.08	0.09	1944.8
Cluster1 (10 cams)	-	0.66	0.63	0.21	0.29	31.9
Cluster2 (15 cams)	-	0.31	0.32	0.18	0.14	111.5
Cluster3 (15 cams)	-	0.31	0.31	0.22	0.21	93.4
Aligned (30 cams)	-	0.53	0.44	0.35	0.32	247.5
Full (30 cams)	2	0.28	0.28	0.18	0.19	1776.0
Cluster1 (10 cams)	-	1.57	1.63	1.00	0.79	28.9
Cluster2 (15 cams)	-	0.77	0.76	0.41	0.36	112.2
Cluster3 (15 cams)	-	0.63	0.59	0.46	0.40	92.9
Aligned (30 cams)	-	1.14	0.97	0.74	0.62	245.7

Table 2. Results for synthetic distributed synchronization. $\bar{e_t}$: mean location error. $\hat{e_t}$: median location error. $\bar{e_r}$: mean rotation error. $\hat{e_r}$: median rotation error. Noise is added as percentage onto the cameras. In the noisy cases, only the quadrifocal tensors were synchronized.

We note that we added uniform noise to the quadrifocal tensor estimates. Yet, the algorithm performs better in the corrupted setting,

where some estimates are very accurate, and a smaller portion is very inaccurate. The main point that we would to demonstrate is that the algorithm can be significantly sped up when the viewing graph has obvious clusters without sacrificing too much accuracy. It is clear that the algorithm can be even further sped up when given more cores and computing resources.

11. Detailed Numerical Results

In this section, we include more comprehensive results accompanying the results in the main paper. We report the completion rates and an idea of the runtime in Table 3. We also report the detailed mean location error, median location error, mean rotation error, median rotation error in Tables 4, 5, 6, 7 respectively. Locations are reported in the metrics of their original datasets, and rotations are reported in degrees.

Table 3. Completion rates and runtimes of different methods

Dataset	Completion %	Joint Opt. (s)	TrifocalSync (s)
courtyard (18/38)	33.10	111.2	137.9
electro (12/39)	78.70	30.6	45.13
facade (46/76)	61.98	4226.2	1633.0
kicker (20/30)	31.84	157.3	174.3
meadow (6/14)	42.13	11.7	9.1
office (13/21)	39.92	34.2	53.3
pipes (8/14)	58.79	21.6	27.2
relief_2 (16/20)	**14.31**	64.9	121.6
relief (13/13)	95.21	36.3	75.4
terrace (11/23)	92.53	27.5	54.8
terrains (23/42)	**8.29**	223.1	238.5
CastleP19 (13/19)	**24.46**	45.2	77.24
CastleP30 (19/30)	**28.02**	139.7	174.7
EntryP10 (10/10)	**26.50**	25.0	41.8
FountainP11 (11/11)	76.55	22.8	47.3
HerzP8 (8/8)	80.76	17.6	26.7
HerzP25 (24/25)	**25.38**	312.7	312.8

Table 4. Mean location error by method

Dataset	QuadSync	Joint Opt.	Trifocal Sync	LUD	NRFM	MPLS BATA	MPLS CS
courtyard (18/38)	0.0477	0.0489	0.1753	0.0403	0.1104	0.0711	**0.0234**
electro (12/39)	0.0199	0.0200	**0.0197**	0.0605	0.0257	0.0617	0.0604
facade (46/76)	3.0294	0.0248	0.0222	0.2497	0.2068	0.0984	**0.0158**
kicker (20/30)	**0.0078**	**0.0078**	0.0204	0.0366	0.0246	0.1934	0.0178
meadow (6/14)	**0.0401**	0.1071	1.4341	0.6408	0.2207	1.8229	0.0691
office (13/21)	0.0039	**0.0037**	0.0151	0.0262	0.0187	0.0166	0.0086
pipes (8/14)	0.0192	0.0115	0.0344	0.1097	0.1058	0.1379	**0.0085**
relief_2 (16/20)	0.5210	0.5360	0.5554	0.8624	0.8526	0.2547	**0.0158**
relief (13/13)	**0.0020**	**0.0020**	**0.0020**	0.0043	0.0047	0.1076	0.0052
terrace (11/23)	**0.0096**	**0.0096**	0.0103	0.0165	0.0150	0.0163	0.0147
terrains (23/42)	0.4631	0.4335	0.0797	0.0092	0.0149	0.0326	**0.0075**
CastleP19 (13/19)	0.5130	0.4921	7.6932	0.2605	0.5965	1.3871	**0.1094**
CastleP30 (19/30)	0.1629	0.1627	6.4738	0.1132	0.1683	0.4945	**0.0687**
EntryP10 (10/10)	**0.0007**	**0.0007**	0.0654	0.0957	0.2913	0.1235	0.0897
FountainP11 (11/11)	**0.0002**	**0.0002**	0.0098	0.0103	0.0223	0.2125	0.0095
HerzP25 (24/25)	**0.0025**	0.0026	0.2187	0.0647	0.1114	9.2329	0.1668
HerzP8 (8/8)	**0.0010**	**0.0010**	0.0180	0.0522	0.2299	0.5766	0.0375

Table 5. Median location error by method

Dataset	QuadSync	Joint Opt.	Trifocal Sync	LUD	NRFM	MPLS BATA	MPLS CS
courtyard	0.0307	0.0324	0.0947	0.0308	0.0668	0.0598	**0.0194**
electro	0.0151	0.0152	0.0133	0.0256	**0.0081**	0.0221	0.0181
facade	2.5853	0.0135	0.0140	**0.0108**	0.0124	0.0253	0.0128
kicker	0.0068	**0.0063**	0.0173	0.0090	0.0110	0.0788	0.0101
meadow	0.0390	0.0922	1.1006	0.2374	0.1285	0.7635	**0.0358**
office	0.0028	**0.0027**	0.0090	0.0035	0.0047	0.0043	0.0033
pipes	0.0139	0.0075	0.0212	0.0390	0.0506	0.1330	**0.0046**
relief_2	0.4392	0.4737	0.5204	0.7633	0.7852	0.2065	**0.0072**
relief	**0.0017**	**0.0017**	**0.0017**	0.0048	0.0048	0.0052	0.0049
terrace	**0.0091**	**0.0091**	0.0104	0.0173	0.0172	0.0164	0.0159
terrains	0.4442	0.4052	0.0609	**0.0065**	0.0114	0.0165	0.0070
CastleP19	0.5502	0.4944	7.2040	0.1694	0.4583	1.1433	**0.0913**
CastleP30	0.1145	0.1169	5.6921	0.0698	0.1536	0.1670	**0.0485**
EntryP10	**0.0006**	**0.0006**	0.0514	0.0851	0.3107	0.1058	0.0571
FountainP11	**0.0002**	**0.0002**	0.0079	0.0086	0.0166	0.1889	0.0085
HerzP25	**0.0009**	0.0010	0.1876	0.0463	0.0849	0.1525	0.0608
HerzP8	**0.0007**	**0.0007**	0.0147	0.0545	0.0808	0.4254	0.0358

Table 6. Mean rotation error by method

Dataset	QuadSync	Joint Opt.	Trifocal Sync	LUD	NRFM	MPLS
courtyard	0.3059	0.3295	1.2398	0.1886	0.1886	**0.1056**
electro	0.1022	0.1020	0.1208	0.0722	0.0722	**0.0603**
facade	2.1185	0.1131	0.1239	**0.0442**	**0.0442**	0.0506
kicker	0.2063	0.2028	0.2986	**0.1162**	**0.1162**	0.2697
meadow	0.2856	0.6100	12.4079	**0.1403**	**0.1403**	0.1861
office	0.1653	0.1449	0.5749	0.0674	0.0674	**0.0547**
pipes	0.5901	0.1785	3.0799	0.1516	0.1516	0.1546
relief_2	32.6666	49.3035	52.5503	45.0175	45.0175	**0.1049**
relief	**0.0791**	**0.0791**	0.0792	0.1080	0.1080	0.1077
terrace	0.0796	0.0796	0.0812	0.0728	0.0728	**0.0693**
terrains	11.5185	11.1041	3.7660	0.2027	0.2027	**0.1918**
CastleP19	30.4922	31.9495	39.5514	0.7604	0.7604	**0.2795**
CastleP30	15.8584	14.8579	10.6018	0.1937	0.1937	**0.1168**
EntryP10	**0.0822**	**0.0822**	0.1489	0.2993	0.2993	0.3086
FountainP11	0.0734	0.0734	0.0912	0.0517	0.0517	**0.0508**
HerzP25	**0.2271**	0.2300	1.1558	0.2879	0.2879	0.7712
HerzP8	**0.0997**	**0.0997**	0.1167	0.4717	0.4717	0.3462

Table 7. Median rotation error by method

Dataset	QuadSync	Joint Opt.	Trifocal Sync	LUD	NRFM	MPLS
courtyard	0.2144	0.2283	1.2651	0.1909	0.1909	**0.1234**
electro	0.0883	0.0888	0.1186	**0.0630**	**0.0630**	0.0638
facade	0.9737	0.0731	0.0812	**0.0449**	**0.0449**	0.0457
kicker	0.1735	0.1701	0.2877	**0.1146**	**0.1146**	0.1650
meadow	0.2002	0.5049	7.3067	**0.0902**	**0.0902**	0.1337
office	0.0994	0.1070	0.2807	0.0717	0.0717	**0.0703**
pipes	0.2937	**0.1131**	1.7915	0.1310	0.1310	0.1312
relief_2	7.9110	12.0705	19.4790	38.7239	38.7239	**0.1039**
relief	0.0880	0.0880	**0.0849**	0.1173	0.1173	0.1157
terrace	**0.0860**	**0.0860**	0.0885	0.0962	0.0962	0.0916
terrains	2.6535	2.3353	2.6054	**0.2280**	**0.2280**	0.2567
CastleP19	6.7486	8.3131	7.7256	0.5380	0.5380	**0.2186**
CastleP30	7.9894	7.8164	4.2029	0.1520	0.1520	**0.0903**
EntryP10	**0.0810**	**0.0810**	0.1173	0.2840	0.2840	0.2375
FountainP11	0.0724	0.0724	0.0884	0.0559	0.0559	**0.0448**
HerzP25	**0.1500**	0.1584	0.7872	0.2564	0.2564	0.4064
HerzP8	**0.0813**	**0.0813**	0.1107	0.4824	0.4824	0.3354