A matter of notation: several uses of the Kronecker product in 3-D Computer Vision

Andrea Fusiello

Dipartimento di Informatica, University of Verona
Strada Le Grazie 15, 37134 Verona, Italy.

Abstract

This work presents a number of cases in Computer Vision where the introduction of the Kronecker product allows more elegant and compact derivations. We hold that a clear notation can enlighten properties and catalyze reasoning. In particular we introduce the *trifocal matrix* that allows to express the trilinear constraints among three views by using the familiar matrix algebra.

Key words: Kronecker product, trifocal matrix, camera calibration, exterior orientation, trifocal geometry

1 Introduction

The interest in the Kronecker product has grown recently, as witnessed by C. Van Loan [1]:

> "The Kronecker product has a rich and very pleasing algebra that supports a wide range of fast, elegant, and practical algorithms. Several trends in scientific computing suggest that this important matrix operation will have an increasingly greater role to play in the future"

In Computer Vision (CV), however, the Kronecker product appeared only sporadically and has not been widely used. One of the first appearances is in [2], where it is used mainly to compute derivatives of matrix functions [3]. For the same purpose it has been exploited later in [4]. In [5,6] the Kronecker product arises in the study of the pre-conditioning of the eight-point-algorithm [7]. In [8] it is used in the context of non-rigid structure from motion. Albeit this sporadic appearances, the Kronecker product has not gained the attention that it probably deserves.

First we will describe the Kronecker product and some related matrix algebra tools. Then we will apply these tools to the derivation of some classical linear algorithm in CV, as the eight-point-algorithm and the Direct Linear Transform (DLT). Then we will re-derive the Zhang's calibration method and the Fiore's algorithm for exterior orientation. In all these cases the use of Kronecker product and related tools yields a compact derivation, where the matrices never need to be expanded in terms of their entries. This allows to reason about global properties of matrices – such as the rank.

The alternative derivations that we provide eventually attain to the same equations of the respective original algorithms. Hence, we refer the reader to the relevant papers for the discussion of their numerical properties.

In the last part we will introduce the *trifocal matrix* that –thanks to the Kronecker product – enables to express the trilinear constraints among three views with matrix algebra. Avoiding the tensorial notation is a great benefit in teaching, because in a typical CV course the exposition of tensor algebra is functional to the trifocal geometry only, hence it constitutes a substantial overhead. Moreover it is fairly unpalatable to the students, who, in our experience, are more proficient with the more familiar matrix algebra. All the previous attempts to avoid the tensorial notation [9,10] sacrifices compactness, meaning that there is not a single algebraic object that "represents" the trilinearity, as our trifocal matrix does instead.

2 Some matrix tools

This section develops some matrix tools related to the Kronecker product that will prove useful in the rest of the paper. Further readings on this topic are [11,3].

2.1 Kronecker product

Let A be a $m \times n$ matrix and B a $p \times q$ matrix. The *Kronecker product* of A and B is the $mp \times nq$ matrix defined by

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix}. \tag{1}$$

The Kronecker product is defined for any pair of matrices A and B. It is associative and distributive with respect to matrix sum and product, but it is not commutative. The transpose of a Kronecker product is $(A \otimes B)^T = A^T \otimes B^T$.

A very important property concerns the eigenvalues of the Kronecker product. Let $A = SLS^{-1}$ and $B = TMT^{-1}$ be the Shur's decomposition of A and B respectively, where L and M are upper triangular matrices whose diagonal elements are the eigenvalues of A and B respectively, and S and T are unitary matrices. Thus,

$$A \otimes B = (SLS^{-1}) \otimes (TMT^{-1}) = (S \otimes T)(L \otimes M)(S^{-1} \otimes T^{-1}). \tag{2}$$

Since $A \otimes B$ and $L \otimes M$ are similar and the latter is upper triangular, the eigenvalues of $A \otimes B$ are the diagonal elements of $L \otimes M$. This implies that

$$\operatorname{rank}(A \otimes B) = \operatorname{rank}(A) \operatorname{rank}(B). \tag{3}$$

2.2 Vectorization

The *vectorization* of a matrix is a linear transformation which converts the matrix into a column vector. Specifically, the vectorization of the matrix A, denoted by $\operatorname{vec}(A)$, is the vector obtained by stacking the columns of A one underneath the other.

The basic connection between the vec operator and the Kronecker product is

$$\operatorname{vec}(\mathbf{a}\mathbf{b}^T) = \mathbf{b} \otimes \mathbf{a} \tag{4}$$

for any column vectors $\mathbf{a}$ and $\mathbf{b}$. The generalization of this is the following important property:

$$\operatorname{vec}(AXB) = (B^T \otimes A) \operatorname{vec}(X) \tag{5}$$

for matrices A, B, X of compatible dimensions. This will be useful for

- pulling the unknown X out of a matrix equation;
- expressing bilinear (and multilinear) forms.

Sometimes, when dealing with symmetric matrices, one wants to vectorize only its unique elements. The *half-vectorization*, $\operatorname{vech}(A)$, of a symmetric $n \times n$ matrix A is the $n(n+1)/2 \times 1$ column vector obtained by vectorizing only the lower triangular part of A. The *duplication matrix* D_n is the unique $n^2 \times n(n+1)/2$ matrix which, transforms $\operatorname{vech}(A)$ into $\operatorname{vec}(A)$: $D_n \operatorname{vech}(A) = \operatorname{vec}(A)$.

2.3 Vector transposition

The *vector transposition* generalizes both vectorization and transposition [12,13]. The p-wise vector-transposition of a $m \times n$ matrix A, denoted by $A^{(p)}$, is obtained by doing a block-transposition of A where blocks are column vectors of p elements (p must divide m, the number of rows of A).

Basic properties are:

$$A^{(1)} = A^T \tag{6}$$

$$A^{(m)} = \mathrm{vec}(A) \tag{7}$$

$$A^{(p)(p)} = A \tag{8}$$

Hence, $\mathrm{vec}(A)^{(m)} = A^{(m)(m)} = A$. The m-wise vector transposition reshapes the column vector $\mathrm{vec}(A)$ back into the matrix A.

3 The eight-point algorithm

A number of 2D-2D point correspondences $\mathbf{m}_\ell^i \leftrightarrow \mathbf{m}_r^i$ (in homogeneous coordinates) is given, and we are required to find the fundamental matrix F that links corresponding points in the bilinear form:

$$\mathbf{m}_r^T F \mathbf{m}_\ell = 0. \tag{9}$$

The *eight-point algorithm* [7] exploits Equation (9) to linearly compute F. Using the Kronecker product and Eq. (5), the derivation of the linear system of equations is particularly easy and elegant, given that one never needs to explode matrices into components [1].

$$\mathbf{m}_r^T F \mathbf{m}_\ell = 0 \iff \mathrm{vec}(\mathbf{m}_r^T F \mathbf{m}_\ell) = 0 \iff (\mathbf{m}_\ell^T \otimes \mathbf{m}_r^T)\,\mathrm{vec}(F) = 0. \tag{10}$$

This is a linear equation in the unknown entries of F. From a set of n point correspondences, we obtain a $n \times 9$ coefficient matrix A by stacking up one equation for each correspondence. The solution is the 1-dimensional right null-space of A. Eight points at least are needed, hence the name.

[1] This derivation first appeared in [2].

4 The Direct Linear Transform algorithm

The *Direct Linear Transform* (DLT) algorithm [10] solves – with small variations – two different problems:

- Camera calibration (or *resection*);
- Homography estimation.

In this section the reader will appreciate the use of the Kronecker notation not only for its compactness, but also because of its rank property (Eq. (3)).

4.1 Camera calibration

A number of 2D-3D point correspondences $\mathbf{m}_i \leftrightarrow \mathbf{M}_i$ (in homogeneous coordinates) is given, and we are required to find a camera matrix P such that

$$\mathbf{m}_i \simeq P\mathbf{M}_i \quad \text{for all } i \tag{11}$$

where the symbol $\simeq$ means equality up to a scale. In order to get rid of this unknown scale factor, the equation can be rewritten in terms of the cross product as

$$\mathbf{m}_i \times P\mathbf{M}_i = \mathbf{0}. \tag{12}$$

Using the properties of the Kronecker product (namely Eq. (5)) we derive:

$$\begin{aligned} \mathbf{m}_i \times P\mathbf{M}_i = \mathbf{0} &\iff [\mathbf{m}_i]_\times P\mathbf{M}_i = \mathbf{0} \iff \\ \operatorname{vec}([\mathbf{m}_i]_\times P\mathbf{M}_i) = \mathbf{0} &\iff (\mathbf{M}_i^T \otimes [\mathbf{m}_i]_\times) \operatorname{vec}(P) = \mathbf{0} \end{aligned} \tag{13}$$

where $[\mathbf{t}]_\times$ is the skew-symmetric matrix such that $\mathbf{t} \times \mathbf{x} = [\mathbf{t}]_\times \mathbf{x}$ for any vector $\mathbf{x}$.

Although there are three equations, only two of them are linearly independent: Indeed, the rank of $(\mathbf{M}_i^T \otimes [\mathbf{m}_i]_\times)$ is two because it is the Kronecker product of a rank-1 matrix by a a rank-2 matrix. From a set of n point correspondences, we obtain a $2n \times 12$ coefficient matrix A by stacking up two equations for each correspondence. The solution is the 1-dimensional right null-space of A. At least 11 equations are needed ($5\frac{1}{2}$ points).

4.2 Homography estimation

A number of 2D-2D point correspondences $\mathbf{m}_r^i \leftrightarrow \mathbf{m}_\ell^i$ (in homogeneous coordinates) is given, and we are required to find the homography matrix H such

that

$$\mathbf{m}_r^i \simeq H\mathbf{m}_\ell^i \quad \text{for all } i \tag{14}$$

The equation (we drop the index i for simplicity) can be rewritten in terms of the cross product as

$$\mathbf{m}_r \times H\mathbf{m}_\ell = \mathbf{0} \tag{15}$$

As we did before, we exploit the properties of the Kronecker product and the vec operator to transform this into a null-space problem and then derive a linear solution:

$$\mathbf{m}_r \times H\mathbf{m}_\ell = \mathbf{0} \iff [\mathbf{m}_r]_\times H\mathbf{m}_\ell = \mathbf{0} \iff$$
$$\mathrm{vec}([\mathbf{m}_r]_\times H\mathbf{m}_\ell) = \mathbf{0} \iff (\mathbf{m}_\ell^T \otimes [\mathbf{m}_r]_\times)\,\mathrm{vec}(H) = \mathbf{0}$$

By the same token as before we conclude that the matrix $(\mathbf{m}_\ell^T \otimes [\mathbf{m}_r]_\times)$ has rank two, hence only two equations out of three are linearly independent. From a set of n point correspondences, we obtain a $2n \times 9$ coefficient matrix A by stacking up two equations for each correspondence. The solution is the 1-dimensional right null-space of A. Four points at least are needed.

5 Zhang's internal calibration

Here we will re-derive the core of Zhang's calibration algorithm [14], i.e., the procedure for computing the internal parameters of a camera starting from world-image homographies.

Several images of a known planar pattern are available, and it is assumed that correspondences between image points and 3-D points on the planar pattern have been established in each view. We are required to find the camera's internal parameters matrix K.

It is easy to see that for a camera $P = K[R|\mathbf{t}]$ the homography between a world plane at $z = 0$ and the image is

$$H \simeq K[\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}] \tag{16}$$

where $\mathbf{r}_i$ are the column of the rotation matrix R. The homography H is computed from correspondences between four or more known world points and their images. Writing $H = [\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3]$, from the previous equation we derive:

$$\mathbf{r}_1 = \lambda K^{-1}\mathbf{h}_1 \tag{17}$$
$$\mathbf{r}_2 = \lambda K^{-1}\mathbf{h}_2 \tag{18}$$

where λ is an unknown scale factor. Thanks to the fact that the columns of R are orthonormal, some constraints can be obtained on the intrinsic parameters.

The orthogonality $\mathbf{r}_1^T\mathbf{r}_2 = 0$ gives $\mathbf{h}_1^T(KK^T)^{-1}\mathbf{h}_2 = 0$ or, equivalently

$$\mathbf{h}_1^T\boldsymbol{\omega}\mathbf{h}_2 = 0 \tag{19}$$

where $\boldsymbol{\omega} = (KK^T)^{-1}$. Likewise, the condition on the norm $\mathbf{r}_1^T\mathbf{r}_1 = \mathbf{r}_2^T\mathbf{r}_2$ gives

$$\mathbf{h}_1^T\boldsymbol{\omega}\mathbf{h}_1 = \mathbf{h}_2^T\boldsymbol{\omega}\mathbf{h}_2 \tag{20}$$

Introducing the Kronecker product as usual (Eq. (5)), the last two equations can be rewritten as:

$$(\mathbf{h}_2^T \otimes \mathbf{h}_1^T)\,\mathrm{vec}(\boldsymbol{\omega}) = 0 \tag{21}$$

$$\Big((\mathbf{h}_1^T \otimes \mathbf{h}_1^T) - (\mathbf{h}_2^T \otimes \mathbf{h}_2^T)\Big)\,\mathrm{vec}(\boldsymbol{\omega}) = 0 \tag{22}$$

As $\boldsymbol{\omega}$ is a 3×3 symmetric matrix, its unique elements (the unknowns) are only six. This fact can be neatly taken into account using the vech operator. The above equations are equivalent to:

$$(\mathbf{h}_2^T \otimes \mathbf{h}_1^T)D_3\,\mathrm{vech}(\boldsymbol{\omega}) = 0 \tag{23}$$

$$\Big((\mathbf{h}_1^T \otimes \mathbf{h}_1^T) - (\mathbf{h}_2^T \otimes \mathbf{h}_2^T)\Big)\,D_3\,\mathrm{vech}(\boldsymbol{\omega}) = 0 \tag{24}$$

From a set of n images, we obtain a $2n \times 6$ coefficient matrix A by stacking up two equations for each image. The solution is the 1-dimensional right null-space of A. At least five equations are needed ($2\frac{1}{2}$ images).

6 Exterior orientation

Given a number of 2D-3D point correspondences $\mathbf{m}^i \leftrightarrow \mathbf{M}^i$ (in homogeneous coordinates) and the intrinsic camera parameters K, we are required to find a rotation matrix R and a translation vector $\mathbf{t}$ (which specify attitude and position of the camera) such that:

$$K^{-1}\mathbf{m}^i \simeq [R|\mathbf{t}]\mathbf{M}^i \quad \text{for all } i. \tag{25}$$

The problem can be cast as a camera resection and solved with the DLT algorithm, but the resulting rotation matrix R is not guaranteed to be orthonormal. Hence, Fiore's algorithm [15] is to be preferred, which is linear and produces a rotation matrix that is inherently orthonormal. We will outline it here.

Let us rewrite Eq. (25) by explicitly introducing the depth [2] of $\mathbf{M}^i$, denoted by ζ^i:

$$\zeta^i K^{-1}\mathbf{m}^i = [R|\mathbf{t}]\mathbf{M}^i \quad \text{for all } i. \tag{26}$$

[2] The depth of a point is its distance from the focal plane of the camera.

The core of Fiore's algorithm is the recovery of the unknown depth ζ^i. Let us write Eq. (26) in matrix form:

$$K^{-1}\underbrace{[\zeta^1\mathbf{m}^1, \zeta^2\mathbf{m}^2, \dots \zeta^n\mathbf{m}^n]}_{W} = [R|\mathbf{t}]\underbrace{[\mathbf{M}^1, \mathbf{M}^2, \dots \mathbf{M}^n]}_{M}.$$

Let $r = \text{rank}(M)$. Take its singular value decomposition: $M = UDV^T$ and let V_2 be a matrix composed by the last $n - r$ columns of V, which span the null-space of M. Then, $MV_2 = 0_{3\times(n-r)}$, and also

$$K^{-1}WV_2 = 0_{3\times(n-r)} \tag{27}$$

By vectorizing both sides we get:

$$(V_2^T \otimes K^{-1})\,\text{vec}(W) = \mathbf{0}. \tag{28}$$

Let us observe that [3]:

$$\text{vec}(W) = \begin{bmatrix} \zeta^1\mathbf{m}^1 \\ \zeta^1\mathbf{m}^2 \\ \vdots \\ \zeta^n\mathbf{m}^n \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{m}^1 & 0 & \dots & 0 \\ & & \ddots & \\ 0 & 0 & \dots & \mathbf{m}^n \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} \zeta^1 \\ \vdots \\ \zeta^n \end{bmatrix}}_{\boldsymbol{\zeta}}$$

Hence

$$\Big((V_2^T \otimes K^{-1})D\Big)\,\boldsymbol{\zeta} = \mathbf{0}. \tag{29}$$

From the last equation the depths $\boldsymbol{\zeta}$ can be recovered (up to a scale factor) by solving a null-space problem.

The size of the coefficients matrix is $3(n-r) \times n$, and in order to determine a one-parameter family of solutions, it must have rank $n-1$, hence $3(n-r) \geq n-1$. Therefore, at least $n \geq (3r-1)/2$ points are needed: six point in general position, or four point on a plane, are sufficient.

Now that the left side of Eq. (26) is known, up to a scale factor, we are left with an absolute orientation (with scale) problem. which can be solved linearly with [16]. As a result, the estimated rotation matrix is orthonormal by construction.

[3] This observation is due to P. Fiore, personal communication.

7 The trifocal constraint

We have demonstrated how the Kronecker notation can yield compact and elegant derivations for some Computer Vision algorithm. We shall now demonstrate how the trifocal constraint can be introduced without resorting to trilinear tensors, thanks to the Kronecker product. This is probably the greatest merit of this notation.

Consider a point $\mathbf{M}$ in space projecting to $\mathbf{m}_1$, $\mathbf{m}_2$ and $\mathbf{m}_3$ in the three cameras

$$P_1 = [I|0], \quad P_2 = [A_2|\mathbf{e}_{2,1}], \quad \text{and } P_3 = [A_3|\mathbf{e}_{3,1}]. \tag{30}$$

Let us write the epipolar line of $\mathbf{m}_1$ in the other two views:

$$\zeta_2 \mathbf{m}_2 = \mathbf{e}_{2,1} + \zeta_1 A_2 \mathbf{m}_1 \tag{31}$$
$$\zeta_3 \mathbf{m}_3 = \mathbf{e}_{3,1} + \zeta_1 A_3 \mathbf{m}_1. \tag{32}$$

where ζ_i varies in $\mathbb{R}$ and correspond to the depth of the 3-D point with respect to view i. Consider a line through $\mathbf{m}_2$, represented by $\mathbf{s}_2$; we have $\mathbf{s}_2^T \mathbf{m}_2 = 0$, that substituted in (31) gives:

$$0 = \mathbf{s}_2^T \mathbf{e}_{2,1} + \zeta_1 \mathbf{s}_2^T A_2 \mathbf{m}_1 \tag{33}$$

Likewise, for a line through $\mathbf{m}_3$ represented by $\mathbf{s}_3$ we can write:

$$0 = \mathbf{s}_3^T \mathbf{e}_{3,1} + \zeta_1 \mathbf{s}_3^T A_3 \mathbf{m}_1 \tag{34}$$

After eliminating ζ_1 from Equation (33) and (34) we obtain:

$$0 = (\mathbf{s}_2^T \mathbf{e}_{2,1})(\mathbf{s}_3^T A_3 \mathbf{m}_1) - (\mathbf{s}_3^T \mathbf{e}_{3,1})(\mathbf{s}_2^T A_2 \mathbf{m}_1) \tag{35}$$

and after some re-writing:

$$0 = \mathbf{s}_2^T \left(\mathbf{e}_{2,1} \mathbf{m}_1^T A_3^T - A_2 \mathbf{m}_1 \mathbf{e}_{3,1}^T \right) \mathbf{s}_3 \tag{36}$$

This is the *trifocal constraint*, that links $\mathbf{m}_1$, $\mathbf{s}_2$ (any line through $\mathbf{m}_2$) and $\mathbf{s}_3$ (any line through $\mathbf{m}_3$). This is the same expression found in [9].

Geometrically, the trifocal constraint imposes that the optical ray of $\mathbf{m}_1$ intersects the 3-D line L that projects onto $\mathbf{s}_2$ in the second image and $\mathbf{s}_3$ in the third image (Fig. 1).

A better expression for the trifocal constraint should incorporate all the coefficients of the trilinear form in a single object. The tensor notation [17–19] satisfies this requirement, but it is cumbersome and hardly palatable for non-experts. The Kronecker notation offers an alternative that relies on matrix algebra.

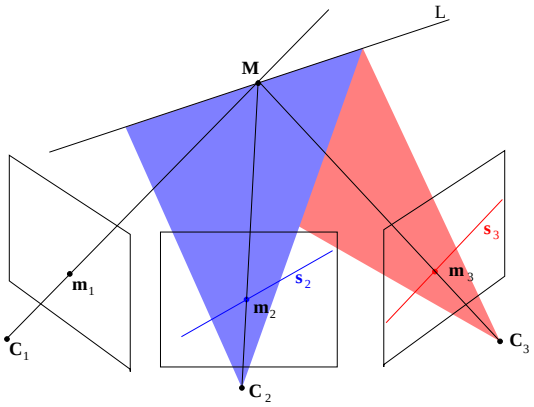

Fig. 1. Two arbitrary lines $\mathbf{s}_2$ and $\mathbf{s}_3$ through corresponding points $\mathbf{m}_2$ and $\mathbf{m}_3$ in the second and third image respectively, define a 3-D line L that must intersect the optical ray of $\mathbf{m}_1$.

Using the properties of the Kronecker product, the trifocal constraint (Eq. (36)) can be written as:

$$\begin{aligned} 0 &= (\mathbf{s}_3^T \otimes \mathbf{s}_2^T)\,\mathrm{vec}\left(\mathbf{e}_{2,1}\mathbf{m}_1^T A_3^T - A_2\mathbf{m}_1\mathbf{e}_{3,1}^T\right) \\ &= (\mathbf{s}_3^T \otimes \mathbf{s}_2^T)\left((A_3 \otimes \mathbf{e}_{2,1})\,\mathrm{vec}(\mathbf{m}_1) - (\mathbf{e}_{3,1} \otimes A_2)\,\mathrm{vec}(\mathbf{m}_1)\right) \\ &= (\mathbf{s}_3^T \otimes \mathbf{s}_2^T)\left((A_3 \otimes \mathbf{e}_{2,1}) - (\mathbf{e}_{3,1} \otimes A_2)\right)\mathbf{m}_1) \\ &= (\mathbf{s}_3^T \otimes \mathbf{s}_2^T)T\mathbf{m}_1 \end{aligned} \tag{37}$$

where T is the 9×3 *trifocal matrix* defined by

$$T = (A_3 \otimes \mathbf{e}_{2,1}) - (\mathbf{e}_{3,1} \otimes A_2) \tag{38}$$

The matrix T encodes the trifocal geometry. Its 27 entries are the coefficient of the trilinear form.

An equivalent formulation of the trifocal constraint that generalizes the expression of a bilinear form (as in Eq. (10)) is obtained by applying Eq. (5) to Eq. (37):

$$(\mathbf{m}_1^T \otimes \mathbf{s}_3^T \otimes \mathbf{s}_2^T)\,\mathrm{vec}(T) = 0. \tag{39}$$

A third equivalent formulation of the trifocal constraint is derived if we look at the vector $T\mathbf{m}_1$ in Eq. (37) as the vectorization of a suitable matrix. This is easy to write thanks to the vector transposition:

$$0 = (\mathbf{s}_3^T \otimes \mathbf{s}_2^T)T\mathbf{m}_1 \tag{40}$$

$$= (\mathbf{s}_3^T \otimes \mathbf{s}_2^T)\,\mathrm{vec}(T\mathbf{m}_1)^{(3)} \tag{41}$$

$$= \mathbf{s}_2^T(T\mathbf{m}_1)^{(3)}\mathbf{s}_3 \tag{42}$$

7.0.1 Relationship with the trifocal tensor.

The Kronecker notation and the tensorial notation are deeply related, as both represents multilinear forms. To draw this relationship in the case of the trifocal geometry, let us expand the trifocal matrix into its columns $T = [\mathbf{t}_1|\mathbf{t}_2|\mathbf{t}_3]$ and $\mathbf{m}_1$ into its components $\mathbf{m}_1 = [u, v, w]^T$. Then, thanks to the linearity of the vector transposition:

$$(T\mathbf{m}_1)^{(3)} = ([\mathbf{t}_1|\mathbf{t}_2|\mathbf{t}_3]\mathbf{m}_1)^{(3)} = (u\mathbf{t}_1 + v\mathbf{t}_2 + w\mathbf{t}_3)^{(3)} = u\mathbf{t}_1^{(3)} + v\mathbf{t}_2^{(3)} + w\mathbf{t}_3^{(3)} \quad (43)$$

This implies that $(T\mathbf{m}_1)^{(3)}$ can be seen as the linear combination of the matrices $\mathbf{t}_1^{(3)}, \mathbf{t}_2^{(3)}, \mathbf{t}_3^{(3)}$ with the components of $\mathbf{m}_1$ as coefficients. Therefore, the action of the trilinear form Eq. (42) is to first combine matrices $\mathbf{t}_1^{(3)}, \mathbf{t}_2^{(3)}, \mathbf{t}_3^{(3)}$ according to $\mathbf{m}_1$, then combine the columns of the resulting matrix according to $\mathbf{s}_3$ and finally to combine the elements of the resulting vector according to $\mathbf{s}_2$, to obtain a scalar.

The $3\times3\times3$ array $\mathcal{T}$ obtained by stacking the three 3×3 matrices $\mathbf{t}_1^{(3)}, \mathbf{t}_2^{(3)}, \mathbf{t}_3^{(3)}$ is the *trifocal tensor*.

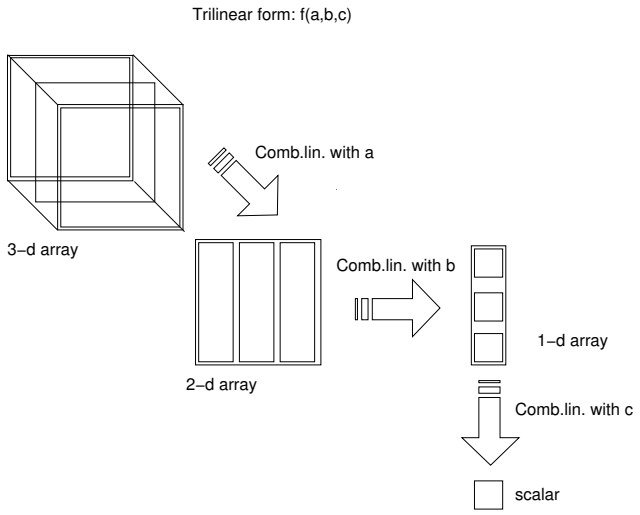

Fig. 2. Action of a trilinear form f(a,b,c) represented by a tensor.

7.0.2 Trifocal constraint for lines.

Consider a line $\mathbf{L}$ in space projecting to $\mathbf{s}_1$, $\mathbf{s}_2$ and $\mathbf{s}_3$ in the three cameras. The trifocal constraint must hold for any point $\mathbf{m}_1$ contained in the line $\mathbf{s}_1$:

$$(\mathbf{s}_3^T \otimes \mathbf{s}_2^T)T\mathbf{m}_1 = 0 \quad \forall \mathbf{m}_1 : \mathbf{s}_1^T\mathbf{m}_1 = 0 \quad (44)$$

hence

$$\mathbf{s}_1^T = (\mathbf{s}_3^T \otimes \mathbf{s}_2^T)T \quad (45)$$

This is the trifocal constraint for lines, which also allows direct line transfer: if $\mathbf{s}_3$ and $\mathbf{s}_2$ are two lines in the third and second view respectively, the image $\mathbf{s}_1$ in the first view of the line in space determined by $\mathbf{s}_2$ and $\mathbf{s}_3$ is obtained by means of the trifocal matrix.

7.0.3 Trifocal constraints for points.

As we already pointed out, $\mathbf{s}_2$ is *any* line through $\mathbf{m}_2$, and $\mathbf{s}_3$ is *any* line through $\mathbf{m}_3$. Each row of $[\mathbf{m}_2]_\times$ (resp. $[\mathbf{m}_3]_\times$) represents a line through $\mathbf{m}_2$ (resp. $\mathbf{m_3}$), because $[\mathbf{m}_2]_\times \mathbf{m}_2 = 0$. Hence, we can rewrite Eq. (36) as:

$$[\mathbf{m}_2]_\times \left(\mathbf{e}_{2,1}\mathbf{m}_1^T A_3^T - A_2\mathbf{m}_1\mathbf{e}_{3,1}^T\right)[\mathbf{m}_3]_\times = 0_{3\times 3}. \tag{46}$$

which is a compact way of writing nine constraints, only four of which are independent, because a point is determined by two lines. Hence, the trifocal constraints for three points writes:

$$([\mathbf{m}_3]_\times \otimes [\mathbf{m}_2]_\times) T\mathbf{m}_1 = \mathbf{0}. \tag{47}$$

Or, equivalently

$$(\mathbf{m}_1^T \otimes [\mathbf{m}_3]_\times \otimes [\mathbf{m}_2]_\times)\,\mathrm{vec}(T) = \mathbf{0} \tag{48}$$

This equation can be used to recover T (likewise we did for F). The coefficient matrix is a 9×27 matrix; its rank is four, being the Kronecker product of a vector by a rank-2 matrix by a rank-2 matrix. Therefore, every triplet $\{\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3\}$ of corresponding points gives four linear independent equations. Seven triplets determine the 27 entries of T.

It is customary, with linear algorithms, to perform data normalization, consisting of a suitable affine transformation of the points [20]. If points in image i are transformed by the affine matrix H_i, taking into account that lines are transformed by H_i^{-1}, Eq. (39) re-writes:

$$\begin{aligned} 0 &= (\mathbf{s}_3^T H_3^{-1} \otimes \mathbf{s}_2^T H_2^{-1})\hat{T} H_2 \mathbf{m}_1 && (49)\\ &= (\mathbf{s}_3^T \otimes \mathbf{s}_2^T)(H_3^{-1} \otimes H_2^{-1})\hat{T} H_2 \mathbf{m}_1 && (50) \end{aligned}$$

Hence $T = (H_3^{-1} \otimes H_2^{-1})\hat{T} H_2$.

7.0.4 Point transfer.

The point version of Eq. (42) is:

$$[\mathbf{m}_2]_\times (T\mathbf{m}_1)^{(3)} [\mathbf{m}_3]_\times = 0_{3\times 3}. \tag{51}$$

Let $\mathbf{s}_2^T$ be a row of $[\mathbf{m}_2]_\times$, then

$$\left(\mathbf{s}_2^T (T\mathbf{m}_1)^{(3)}\right) [\mathbf{m}_3]_\times = \mathbf{0} \tag{52}$$

This implies that the transpose of the leftmost term in parentheses (which is a 3-D vector) belongs to the kernel of $[\mathbf{m}_3]_\times$, which is equal to $\mathbf{m}_3$ (up to a scale factor) by construction. Hence

$$\mathbf{m}_3 \simeq (T\mathbf{m}_1)^{(3)^T} \mathbf{s}_2 \tag{53}$$

This is the point transfer equation: if $\mathbf{m}_1$ and $\mathbf{m}_2$ are conjugate points in the first and second view respectively, the position of the conjugate point $\mathbf{m}_3$ in the third view is computed by means of the trifocal matrix.

8 Conclusions

We have shown some applications of the Kronecker notation to 3-D Computer Vision problems. We argued that this compact notation, especially in the case of the trifocal constraint, can be a practical aid for teaching and a fruitful tool for reasoning about the properties of the matrices that are involved.

Acknowledgments

Michela Farenzena read the draft and her comments helped to improve the presentation.

References

[1] C. Van Loan, The ubiquitous Kronecker product, J. Comput. Appl. Math 123 (1-2) (2000) 85–100.

[2] P. R. S. Mendonça, Multiview geometry: Profiles and self-calibration, Ph.D. thesis, University of Cambridge, Cambridge, UK (May 2001).

[3] J. R. Magnus, H. Neudecker, "Matrix Differential Calculus with Applications in Statistics and Econometrics", revised Edition, John Wiley & Sons, 1999.

[4] A. Fusiello, A. Benedetti, M. Farenzena, A. Busti, Globally convergent autocalibration using interval analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (12) (2004) 1633–1638.

[5] W. Chojnacki, M. Brooks, A. van den Hengel, D. Gawley, Revisiting Hartley's normalized eight-point algorithm, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (9) (2003) 1172–1177.

[6] E. Izquierdo, V. Guerra, Estimating the essential matrix by efficient linear techniques, IEEE Transactions on Circuits and Systems for Video Technology 13 (9) (2003) 925–935.

[7] R. I. Hartley, Estimation of relative camera position for uncalibrated cameras, in: Proceedings of the European Conference on Computer Vision, Santa Margherita L., 1992, pp. 579–587.

[8] M. Brand, A direct method for 3D factorization of nonrigid motion observed in 2D, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, Washington, DC, USA, 2005, pp. 122–128.

[9] Y. Ma, S. Soatto, J. Kosecka, S. S. Sastry, An Invitation to 3-D Vision, Springer, 2003.

[10] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd Edition, Cambridge University Press, 2003.

[11] R. Horn, C. Johnson, Topics in Matrix Analysis, Cambridge University Press, 1994.

[12] D. Marimont, B. Wandell, Linear models of surface and illuminant spectra, J. Opt. Soc. Am. A 9 (11) (1992) 1905–1913.

[13] T. Minka, Old and new matrix algebra useful for statistics, MIT Media Lab note, http://research.microsoft.com/ minka/papers/matrix/ (2000).

[14] Z. Zhang, A flexible new technique for camera calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (11) (2000) 1330–1334.

[15] P. D. Fiore, Efficient linear solution of exterior orientation., IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (2) (2001) 140–148.

[16] K. Kanatani, Geometric Computation for Machine Vision, Oxford University Press, 1993.

[17] R. Hartley, Lines and Points in Three Views and the Trifocal Tensor, International Journal of Computer Vision 22 (2) (1997) 125–140.

[18] A. Shashua, M. Werman, Trilinearity of three perspective views and its associated tensor, in: Proceedings of the International Conference on Computer Vision, 1995, pp. 920–925.

[19] A. Shashua., Trilinear tensor: The fundamental construct of multiple-view geometry and its applications., in: International Workshop on Algebraic Frames For The Perception Action Cycle (AFPAC), Kiel Germany, 1997.

[20] R. I. Hartley, In defence of the 8-point algorithm, in: Proceedings of the International Conference on Computer Vision, 1995, pp. 1064–1071.