Tensor Method for Constructing 3D Moment Invariants

Tomáš Suk and Jan Flusser

Institute of Information Theory and Automation of the ASCR
{suk,flusser}@utia.cas.cz

Abstract. A generalization from 2D to 3D of the tensor method for derivation of both affine invariants and rotation, translation and scaling (TRS) invariants is described. The method for generation of the 3D TRS invariants of higher orders is automated and experimentally tested.

Keywords: Recognition, tensor, moment, rotation, invariant, 3D.

1 Introduction

Pattern recognition of objects in two-dimensional (2D) images has been an important part of image analysis for many years. The images are often geometrically distorted; the distortion of a flat scene can be modeled as a combination of translation, rotation and scaling (TRS) in the case of a scanning device parallel to the scene and as a projective transformation in the opposite case.

An efficient approach to the recognition of deformed objects is using certain features that do not vary in the transformation; we call them *invariants*. Thus, TRS invariants can be applied to the recognition of objects distorted by TRS, and projective invariants to the recognition of objects distorted by the projective transform. Since it is difficult to derive global projective invariants describing an entire object, the projective transform is often approximated by an affine transformation. If the distance of the scanned scene and the camera is large, this approximation is accurate enough.

Recently, the scanning devices of 3D objects (computer tomography (CT), magnetic resonance imaging (MRI), rangefinders, etc.) become more and more affordable, which arises the need of recognition of 3D objects and thus the need of having 3D invariants. One of the most popular family of 3D invariants is based on image moments. While moment invariants in 2D have been studied extensively for decades (see [3] for a survey and [7] for the latest results), the theory of 3D moment invariants has not been fully explored. The first attempts to derive 3D rotation moment invariants are relatively old (see e.g. [6], [4]), but the generalization to higher moment orders has not been reported as it is rather complicated.

The topic of this paper is generalization of the tensor method for deriving rotation moment invariants in 3D. The main contribution is not only finding a closed-form solution for 3D rotation and an affine invariant but principally an algorithm, which generates all invariants up to a given order.

A. Berciano et al. (Eds.): CAIP 2011, Part II, LNCS 6855, pp. 212–219, 2011.

2 Moments and Tensors

Geometric moments of image f (2D or 3D) are defined as

$$\begin{aligned} m_{pq} &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)\ \mathrm{d}x\ \mathrm{d}y, \\ m_{pqr} &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q z^r f(x,y,z)\ \mathrm{d}x\ \mathrm{d}y\ \mathrm{d}z. \end{aligned} \tag{1}$$

The sum of the indices is called the *order* of the moment. To provide the translation invariance, we often use the *central geometric moments*

$$\mu_{pqr} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-x_c)^p (y-y_c)^q (z-z_c)^r f(x,y,z)\ \mathrm{d}x\ \mathrm{d}y\ \mathrm{d}z, \tag{2}$$

where $x_c = m_{100}/m_{000}$, $y_c = m_{010}/m_{000}$ and $z_c = m_{001}/m_{000}$ are centroid coordinates of the image $f(x,y,z)$.

We can define a *moment tensor* [1] for using tensor calculus for derivation of moment invariants

$$M^{i_1 i_2 \cdots i_k} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^{i_1} x^{i_2} \cdots x^{i_k} f(x^1, x^2, x^3)\ \mathrm{d}x^1\ \mathrm{d}x^2\ \mathrm{d}x^3, \tag{3}$$

where $x^1 = x$, $x^2 = y$ and $x^3 = z$. If p indices equal 1, q indices equal 2 and r indices equal 3, then $M^{i_1 i_2 \cdots i_k} = m_{pqr}$. The definition in 2D is analogous. The behavior of the moment tensor under an affine transform (in Einstein notation[1]) is

$$\begin{aligned} M^{i_1 i_2 \cdots i_r} &= |J| p^{i_1}_{\alpha_1} p^{i_2}_{\alpha_2} \cdots p^{i_r}_{\alpha_r} \hat{M}^{\alpha_1 \alpha_2 \cdots \alpha_r} \\ \hat{M}^{i_1 i_2 \cdots i_r} &= |J|^{-1} q^{i_1}_{\alpha_1} q^{i_2}_{\alpha_2} \cdots q^{i_r}_{\alpha_r} M^{\alpha_1 \alpha_2 \cdots \alpha_r}, \end{aligned} \tag{4}$$

where p^i_α is the matrix of the direct affine transform and q^i_α is the matrix of the inverse affine transform (without translation). This means that the moment tensor is a relative contravariant tensor with the weight -1.

2.1 Affine Invariants in 2D and in 3D

The tensor method for affine invariants in 2D is described e.g. in [5]: we arrange our measurements into a tensor, multiply the tensors so the number of contravariant indices equals the number of covariant indices and compute the total contraction of the product. Since the moment tensor is purely contravariant, we need to multiply them by some covariant tensors. In such a case, we can use so-called unit polyvectors.

[1] A. Einstein introduced this notation to simplify expressions with n-dimensional coordinates, see e.g. [9] for explanation.

The *unit polyvector* is an antisymmetric tensor over all indices and the component with indices $1, 2, \ldots, n$ equals 1. It can be both covariant, i.e. $\epsilon_{12\cdots n} = 1$ and contravariant, i.e. $\epsilon^{12\cdots n} = 1$. The term *antisymmetric* means that the tensor component changes its sign and preserves its magnitude when interchanging two arbitrary indices. In 2D, it means that (in matrix notation except for the unit polyvector is not multiplied like a matrix)

$$\varepsilon_{i_1 i_2} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \tag{5}$$

Then a proper tensor product can be used for derivation of a relative affine invariant, e.g.

$$M^{ij} M^{kl} \epsilon_{ik} \epsilon_{jl} = 2(m_{20} m_{02} - m_{11}^2).$$

After modification, we obtain an absolute affine invariant

$$I_1^{2D} = (\mu_{20}\mu_{02} - \mu_{11}^2)/\mu_{00}^4.$$

The exponent $\omega = 4$ of μ_{00} equals the number of factors in the tensor product, i.e. the moment tensors and the unit polyvectors. Other example

$$M^{ijk} M^{lmn} M^{opq} M^{rst} \epsilon_{il} \epsilon_{jm} \epsilon_{ko} \epsilon_{lr} \epsilon_{ps} \epsilon_{qt} :$$
$$I_2^{2D} = (-\mu_{30}^2\mu_{03}^2 + 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} - 4\mu_{30}\mu_{12}^3 - 4\mu_{21}^3\mu_{03} + 3\mu_{21}^2\mu_{12}^2)/\mu_{00}^{10}.$$

A question how to generate all relevant tensor products arises. We can employ an idea, that the tensor products can be described by graphs and then generation of all tensor products means generation of all graphs with the corresponding number of nodes and edges; this is a so-called graph method [3].

In 3D, the unit polyvector looks like

$$\varepsilon_{i_1 i_2 1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}, \ \varepsilon_{i_1 i_2 2} = \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \ \varepsilon_{i_1 i_2 3} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{6}$$

There are two differences between 2D and 3D; the unit polyvector has three indices, i.e. we need fewer factors in the tensor product, and each index can have three values 1, 2 and 3. In the sense of Einstein notation, we sum over these three values. Some examples

$$M^{ij} M^{kl} M^{mn} \epsilon_{ikm} \epsilon_{jln} :$$
$$I_1^{3D} = (\mu_{200}\mu_{020}\mu_{002} + 2\mu_{110}\mu_{101}\mu_{011} - \mu_{200}\mu_{011}^2 - \mu_{020}\mu_{101}^2 - \mu_{002}\mu_{110}^2)/\mu_{000}^5.$$

and

$$M^{ijk} M^{lmn} M^{opq} M^{rst} \epsilon_{ilo} \epsilon_{jmr} \epsilon_{kps} \epsilon_{nqt} :$$
$$I_2^{3D} = (\mu_{300}\mu_{003}\mu_{120}\mu_{021} + \mu_{300}\mu_{030}\mu_{102}\mu_{012} + \mu_{030}\mu_{003}\mu_{210}\mu_{201} - \mu_{300}\mu_{120}\mu_{012}^2$$
$$-\mu_{300}\mu_{102}\mu_{021}^2 - \mu_{030}\mu_{210}\mu_{102}^2 - \mu_{030}\mu_{201}^2\mu_{012} - \mu_{003}\mu_{210}^2\mu_{021} - \mu_{003}\mu_{201}\mu_{120}^2$$
$$-\mu_{300}\mu_{030}\mu_{003}\mu_{111} + \mu_{300}\mu_{021}\mu_{012}\mu_{111} + \mu_{030}\mu_{201}\mu_{102}\mu_{111} + \mu_{003}\mu_{210}\mu_{120}\mu_{111}$$
$$+\mu_{210}^2\mu_{012}^2 + \mu_{201}^2\mu_{021}^2 + \mu_{120}^2\mu_{102}^2 - \mu_{210}\mu_{120}\mu_{102}\mu_{012} - \mu_{210}\mu_{201}\mu_{021}\mu_{012}$$
$$-\mu_{201}\mu_{120}\mu_{102}\mu_{021} - 2\mu_{210}\mu_{012}\mu_{111}^2 - 2\mu_{201}\mu_{021}\mu_{111}^2 - 2\mu_{120}\mu_{102}\mu_{111}^2$$
$$+3\mu_{210}\mu_{102}\mu_{021}\mu_{111} + 3\mu_{201}\mu_{120}\mu_{012}\mu_{111} + \mu_{111}^4)/\mu_{000}^8.$$

The idea of the graphs is more complicated in 3D, we cannot use the ordinary graphs, we need so-called three-uniform hypergraphs, where each hyperedge connects three nodes.

2.2 Rotation Invariants in 2D and in 3D

When the transformation in question is not a full affine transform, but is a mere TRS, then a slightly different approach is suitable to yield simpler invariants. So-called *Cartesian tensors* are suitable for derivation of the rotational invariants. The ordinary tensor behaves in the affine transformation according to the rule

$$\hat{T}^{\beta_1,\beta_2,\ldots,\beta_{k_2}}_{\alpha_1,\alpha_2,\ldots,\alpha_{k_1}} = q^{\beta_1}_{j_1} q^{\beta_2}_{j_2} \cdots q^{\beta_{k_2}}_{j_{k_2}} p^{i_1}_{\alpha_1} p^{i_2}_{\alpha_2} \cdots p^{i_{k_1}}_{\alpha_{k_1}} T^{j_1,j_2,\ldots,j_{k_2}}_{i_1,i_2,\ldots,i_{k_1}}, \tag{7}$$

while the Cartesian tensor behaves in the rotation according to the rule

$$\hat{T}_{\alpha_1,\alpha_2,\ldots,\alpha_k} = r_{\alpha_1 i_1} r_{\alpha_2 i_2} \cdots r_{\alpha_k i_k} T_{i_1,i_2,\ldots,i_k}, \tag{8}$$

where r_{ij} is an arbitrary orthonormal matrix. The distinction of the covariant and contravariant tensors has no meaning in the case of the Cartesian tensors and we can perform the total contraction of the moment product without the unit polyvectors.

The simplest example is the total contraction of the second-order moment tensor M_{ii}. In 2D

$$M_{11} + M_{22} : \Phi_1^{2D} = (\mu_{20} + \mu_{02})/\mu_{00}^2,$$

in 3D

$$M_{11} + M_{22} + M_{33} : \Phi_1^{3D} = (\mu_{200} + \mu_{020} + \mu_{002})/\mu_{000}^{5/3}.$$

Another example is $M_{ij}M_{ij}$. In 2D

$$\Phi_2^{2D} = (\mu_{20}^2 + \mu_{02}^2 + 2\mu_{11}^2)/\mu_{00}^4,$$

in 3D

$$\Phi_2^{3D} = (\mu_{200}^2 + \mu_{020}^2 + \mu_{002}^2 + 2\mu_{110}^2 + 2\mu_{101}^2 + 2\mu_{011}^2)/\mu_{000}^{10/3}.$$

The last example of the second order is $M_{ij}M_{jk}M_{ki}$. In 2D it is

$$\Phi_3^{2D} = (\mu_{20}^3 + 3\mu_{20}\mu_{11}^2 + 3\mu_{11}^2\mu_{02} + \mu_{02}^3)/\mu_{00}^6,$$

while in 3D it becomes

$$\begin{aligned}\Phi_3^{3D} = (&\mu_{200}^3 + 3\mu_{200}\mu_{110}^2 + 3\mu_{200}\mu_{101}^2 + 3\mu_{110}^2\mu_{020} + 3\mu_{101}^2\mu_{002} + \mu_{020}^3 \\ &+3\mu_{020}\mu_{011}^2 + 3\mu_{011}^2\mu_{002} + \mu_{002}^3 + 6\mu_{110}\mu_{101}\mu_{011})/\mu_{000}^5.\end{aligned}$$

The scaling normalization is slightly more difficult; the exponent is

$$\omega = \frac{w}{n} + s, \tag{9}$$

where w is the sum of the indices of all moments in one term, s is the number of the moments in one term and n is the dimension.

2.3 Automated Generation of the Rotation Invariants in 3D

We recall the idea of using graphs for constructing invariants. In the case of a 3D rotation, the ordinary graphs, where each edge connects two nodes, are sufficient. These graphs can include self-loops, see e.g. Fig. 1a.

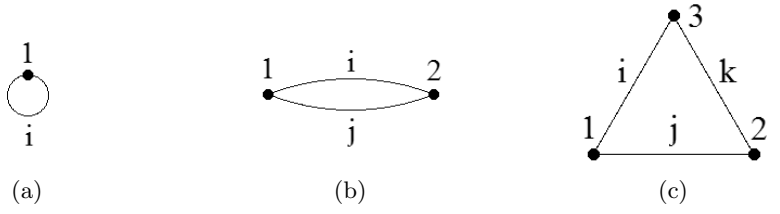

Fig. 1. The generating graphs of (a) both Φ_1^{2D} and Φ_1^{3D}, (b) both Φ_2^{2D} and Φ_2^{3D} and (c) both Φ_3^{2D} and Φ_3^{3D}

An important part of this process is an elimination of the linearly dependent invariants, i.e. zeros, identical invariants, products and linear combinations. The invariants remaining after this elimination are called *irreducible*; those which were eliminated are called *reducible* invariants. It should be noted that irreducibility does not mean independence. There may be polynomial dependencies among irreducible invariants which are not discovered in the elimination algorithms. As will be shown, there must be a large number of them but their identification is an extremely complex problem even in 2D.

The method of finding all irreducible invariants we propose here is a generalization of our method for 2D case [3]. We first generate all possible invariants (graphs) and then, by exhaustive search, eliminate the reducible ones. Everything is carried out on symbolic level, independent of any particular image data. As a result of this algorithm, we obtain closed-form expressions for all irreducible invariants along with automatically generated data for their calculation. Taking into account the computing complexity on the one hand and the capability of our computers on the other, the maximum number of graph edges which are feasible to construct is 8. Table 1 summarizes their numbers.

The first row of the table shows the orders of the invariants, the second row contains the cumulative number of the irreducible invariants up to the given order which were actually constructed by the proposed algorithm (note that there is no guarantee that all existing irreducible invariants were found because of the limitation to maximum of 8 graph edges), and the third row contains the theoretical maximum number of the independent invariants. The number of the independent invariants was estimated as a difference between the number of moments and the number of degrees of freedom of the transformation, i.e. $\binom{t+3}{3} - 7$ up to the order t in our case. As previously mentioned, currently we are not able to systematically find these independent invariants (i.e. to identify polynomial dependencies among irreducible invariants). This extremely difficult task will be a subject of future research.

Table 1. The numbers of the 3D irreducible and independent rotation invariants

Order	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
irred.	3	42	242	583	840	1011	1098	1142	1164	1174	1180	1182	1184	1184	1185
indep.	3	13	28	49	77	113	158	213	279	357	448	553	673	809	962

We generated explicit forms of all 1185 invariants, they are available on our website [2]. However, the corresponding pdf file contains more than 10 000 pages.

3 Numerical Experiment

The following simple experiment verifies that the constructed irreducible invariants are actually invariant to rotation, i.e. they were derived correctly. Moreover, it demonstrates that two similar but different objects have different values of (at least some) invariants.

The experiment was carried out on real data. We used two ancient Greek amphoras scanned using a laser rangefinder from various sides. Consequently all measurements were combined to obtain a 3D binary image of the amphora. Since the rangefinder cannot get inside the amphora, it is considered filled up and closed on the top. To compress the data, the surface of the amphora was divided into small triangles (42 400 and 23 738 triangles, respectively). The amphoras were then represented only by their triangulated surfaces. The test data are shown in Fig. 2. The photographs are for illustration only, no graylevel/color information was used in moment calculation.

(a) (b) (c)

Fig. 2. The amphoras: (a) photo of A1, (b) photo of A2, (c) wire model of the triangulation of A2

For each amphora we generated 10 random rotations and translations of its triangular representation[2] and calculated the values of the first 242 invariants up to the 4th order. Here we have two possibilities as to what moments to use for computing invariants – we can employ either traditional 3D volume moments or *surface moments* [10]. Surface moments are calculated by double integration over the object surface only. We used both approaches.

The maximum relative standard deviation was $3.2 \cdot 10^{-13}$ in the case of invariants computed from the volume moments and $4.5 \cdot 10^{-13}$ for the invariants from the surface moments, which illustrates a perfect invariance in all cases. On the other hand, the maximum relative deviations between two different amphoras A1 and A2 were 1.01 for volume invariants and 1.55 for surface invariants, which proves discriminability – different objects have distinct values of the invariants.

3.1 Computing Moments of Triangulated Objects

Now we explain how the 3D moments (both volume and surface) were actually calculated in this experiment. It would be of course possible to calculate them from definition but since we already have the triangular representation, the moments can be calculated in a more efficient way [8].

We complete each triangle to a tetrahedron such that the new vertex coincides with the coordinate origin. The triangles must have the same orientation with respect to the object, e.g. counterclockwise when seeing from outside to inside of the object. The object volume is then divided into these disjoint tetrahedrons and the volume moment is given as a sum of moments of all tetrahedrons. Using the vertices of the triangulation only, the volume moment is calculated as

$$m_{pqr} = \frac{p!q!r!}{(p+q+r+n)!} \sum_{(k_{ij})\in\mathcal{K}} \frac{\prod_{j=1}^{3}((\sum_{i=1}^{3} k_{ij})!)}{\prod_{i,j=1}^{3}(k_{ij}!)} \sum_{\ell=1}^{N} A_\ell \prod_{i,j=1}^{3} \left(a_{ij}^{(\ell)}\right)^{k_{ij}}, \qquad (10)$$

where N is the number of the triangles, $\mathcal{K}$ is a set of such 3×3 matrices k_{ij} with non-negative integer values that $\sum_{j=1}^{3} k_{1j} = p$, $\sum_{j=1}^{3} k_{2j} = q$ and $\sum_{j=1}^{3} k_{3j} = r$, $a_{ij}^{(\ell)}$ is a matrix of the vertex coordinates of the ℓ-th triangle, i is the number of the coordinate and j is the number of the vertex. $A_\ell = \det\left(a_{ij}^{(\ell)}\right)$, i.e. it is a 6-multiple of the oriented tetrahedron volume and the dimension $n = 3$.

In the case of the surface moments the formula is basically the same except for $A_\ell = \left\| \left(a_{i2}^{(\ell)} - a_{i1}^{(\ell)}\right) \times \left(a_{i3}^{(\ell)} - a_{i1}^{(\ell)}\right) \right\|$ is twice the oriented area of the triangle. The surface moments have different scaling normalization, the dimension $n = 2$ in (9) and in (10).

[2] A more correct way would be to rotate the amphora physically in the capturing device and scan it again in each position. However, this would be extremely costly and the results would be comparable.

4 Conclusion

We have proposed and implemented a tensor method for generation of 3D rotation moment invariants of arbitrary orders. We tested this method on invariants up to the order 16. We constructed 1185 irreducible invariants, a vast majority of them being published for the first time. Our method includes elimination of linearly dependent invariants, but for now does not contain identification of polynomial dependencies among the invariants.

Acknowledgments. Thanks to the grant No. P103/11/1552 of the Czech Science Foundation for financial support. Thanks to Clepsydra (ΚΛΕΨΥΔΡΑ), The Digitization Center of Cultural Heritage in Xanthi, Greece (`http://clepsydra.ipet.gr`) and especially to prof. Christodoulos Chamzas for the models of amphoras.

References

1. Cyganski, D., Orr, J.A.: Object recognition and orientation determination by tensor methods. In: Huang, T.S. (ed.) Advances in Computer Vision and Image Processing, pp. 101–144. JAI Press, Greenwich (1988)
2. Department of Image Processing: 3D rotation moment invariants, `http://zoi.utia.cas.cz/3DRotationInvariants`
3. Flusser, J., Suk, T., Zitová, B.: Moments and Moment Invariants in Pattern Recognition. Wiley, Chichester (2009)
4. Lo, C.H., Don, H.S.: 3-D moment forms: Their construction and application to object identification and positioning. IEEE Transactions on Pattern Analysis and Machine Intelligence 11(10), 1053–1064 (1989)
5. Reiss, T.H.: Recognizing Planar Objects Using Invariant Image Features. LNCS, vol. 676. Springer, Heidelberg (1993)
6. Sadjadi, F.A., Hall, E.L.: Three dimensional moment invariants. IEEE Transactions on Pattern Analysis and Machine Intelligence 2(2), 127–136 (1980)
7. Suk, T., Flusser, J.: Affine moment invariants generated by graph method. Pattern Recognition 44(9), 2047–2056 (2011)
8. Tuzikov, A.V., Sheynin, S.A., Vasiliev, P.V.: Efficient computation of body moments. In: Skarbek, W. (ed.) CAIP 2001. LNCS, vol. 2124, pp. 201–208. Springer, Heidelberg (2001)
9. Wikipedia: Einstein notation, `http://en.wikipedia.org/wiki/Einsteinnotation`
10. Xu, D., Li, H.: 3-D surface moment invariants. In: Proceedings of the 18th International Conference on Pattern Recognition ICPR 2006, pp. 173–176. IEEE Computer Society, Los Alamitos (2006)