Marginal Notes for Hartley & Zisserman's 'Multiple View Geometry'

editor: Leo Dorst

March 17, 2008

Abstract

During our reading club discussions, some clarifying remarks came up. This is a summary of them. In the following HZ stands for Hartley & Zisserman.

1 Introduction – a Tour

1.1 Points at Infinity (page 2)

Points at infinity are a different term for what we would more commonly call *directions*. The statement that two parallel lines intersect at a point at infinity can then be read as the rather trivial: 'two parallel lines have a direction in common'. Two non-parallel lines in the plane have a point in common, we would say they *intersect*. Homogeneous coordinates are just a way of making 'having an element in common' be a single operation, of which the outcome is an element that we would call a *point* if it is of the form $(x, y, 1)^\top$, and a *direction* if it is of the form $(n_1, n_2, 0)^\top$. HZ prefer to call everything a point, and then a direction is a point at infinity.

The points at infinity form a hyperplane (in 2D, a line); we would call this the *horizon*. A projective transformation can reveal this line as a finite line in an image – it is a picture of the horizon, and its points would normally be called *vanishing points* (where parallel lines meet).

1.2 Oriented Projective Geometry (page 2,26)

The introduction of homogeneous coordinates by mapping a point $(x, y)^\top$ to $(kx, ky, k)^\top$ with $k \neq 0$ seems fine. But using the same mapping to map rays loses useful information. A ray travels alongs its carrier line in a certain direction, and if you can keep this information you will be able to distinguish later between things in front of or behind the camera, when a ray hits an object from the outside or the inside, et cetera. It is better for applications to keep track of

the sign of k for the embedding of a ray or line. This gives *oriented projective geometry*, worked out graphically in pleasant detail by Stolfi [2]. The mathematical structure of projective spaces here does not quite match our needs in imaging. HZ do not seem to be aware of this, which is a pity.

2 Projective Geometry and 2D Transformations

2.1 Algebra and Coordinates (page 25,26)

Using algebra is *not* identical to using coordinates; there is an intermediate level of the algebraic properties with clear geometrical interpretations which HZ ususally skips, but which we will try to highlight. This level is probably best described using geometric algebra [1], but can also be made explicit in the linear algebra notations that HZ use.

2.2 Points and Lines (page 26)

HZ destroy the notational difference between points and lines (hyperplanes in n-D) by denoting both as column vectors. But since they behave differently under projective transformations (a point has $\mathbf{p} \mapsto H\mathbf{p}$, a line goes like $\mathbf{l} \mapsto H^{-\top}\mathbf{l}$), you will need to distinguish them in your software.

Other authors denote points by column vectors, and lines (hyperplanes) by row vectors. That is much more clear, and gets the corresponding transforms automatically correct. This is an annoying example of clumsiness introduced by focusing on coordinates too much, whereas a bit more true algebraic structure would have helped.

2.3 Homogeneous Representations of Mappings (page 34)

It should not surprise us that homogeneous coordinates are useful for imaging, since they correspond precisely to the algebraic structure of what happens when you look at something with a pinhole camera. If you gaze at a 3D point $\mathbf{x} = (x, y, z)^\top$ through a camera with focal length 1, pointed in the $\mathbf{e}_3$-direction, the point maps on the image plane to the 3D point $(x/z, y/z, 1)^\top$ (or at least, the image plane as we draw it in front of the camera, as is common). Working in 2D image coordinates, this is the point $(x', y')^\top = (x/z, y/z)^\top$. So conversely, if we have a 2D plane, we can imagine it as an image plane of a camera of one more dimension, and study all (2+1)-D points that might have mapped onto a given point. These can then be any multiple k of $(x', y', 1)$ - which we could also see as changing the focal length of the pinhole camera to k. Using homogeneous coordinates is literally like looking at an n-D space from an perpendicular extra dimension.

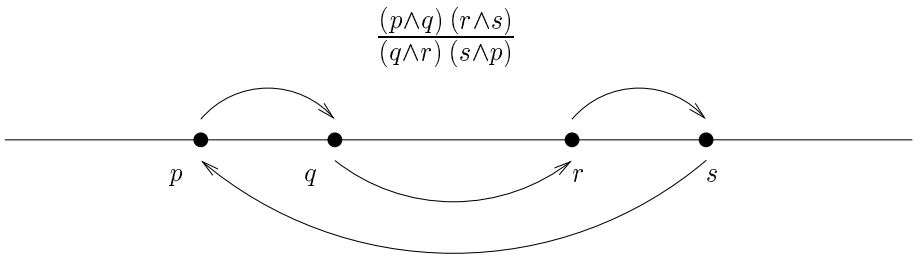


Figure 1: *(From [1]) The combinations of four points taken in the cross ratio. Read the GA expression $p \wedge q$ as the oriented distance from p to q. The directed arcs above the line go into the numerator, the directed arcs below the line into the denominator. The arcs indicate the orientation in which you should measure the distance as positive. Permutation of the points leads to other diagrams, but always the division produces the invariant.*

2.4 Hierarchy of Transformation (page 37 etc)

A transformation matrix indicates in its i-th column where the i-th basis vector goes. Basis vectors are $(1,0,0)^\top$, $(0,1,0)^\top$, $(0,0,1)^\top$. In homogeneous coordinates, we would interpret the first two as direction vectors (since the third component is zero), and the final one as the representation of the point at the origin (since the final component is 1). In this view, a point $(x,y,1)^\top = x\,(1,0,0)^\top + y\,(0,1,0)^\top + (0,0,1)^\top$ is made by moving x in the $\mathbf{e}_1$ direction and y in the $\mathbf{e}_2$-direction, from the origin $(0,0,1)^\top$.

Specifying a transformation that keeps directions as directions and locations (i.e., points) as locations implies that the first column (the image of $\mathbf{e}_1$) should end in 0 (for it maps to a direction), the second in 0 (ditto), and the third should end in 1 (for it maps to a point). That is the form of an affine transformation (and therefore also of a similarity). And if $\mathbf{e}_1 = (1,0)^\top$ rotates to $(\cos\phi, \sin\phi)^\top$, that means the first column of the matrix should be the direction vector $(\cos\phi, \sin\phi, 0)^\top$. The point at the origin goes to $(t_x, t_y)^\top$, so becomes the homogeneous point $(t_x, t_y, 1)^\top$ – which is indeed the third column of the similarity matrix.

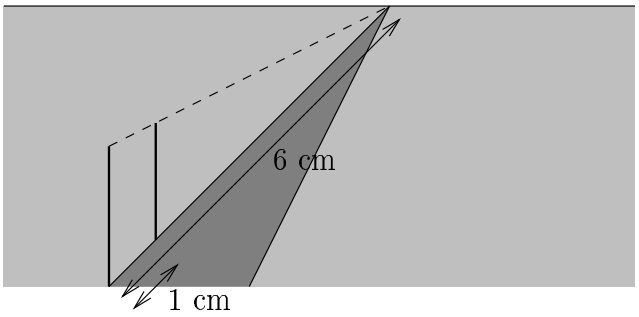
In a projective transformation, the first column may have a last component not equal to 0 – that implies that the direction $(1,0,0)^\top$ has become a finite point. When you take a bird's eye view picture that includes traintracks towards the horizon in the $\mathbf{e}_1$-direction of your planar coordinates, that is precisely what happens: direction (= point at infinity in the original planar scene) becomes finite point (= vanishing point in the picture).

So the last homogeneous coordinate is like a 'point-bit': if it is 1, we represent a point, if it is 0 it is a direction. Of course, it is more than Boolean, its linear change can be given an interpretation in projective transformations.

2.5 Cross Ratio (page 45)

The simplest way to think about a cross ratio is as a ratio of 4 oriented distances, as in Figure 1 (from [1]). You can really use this constructively. Here is an exercise from [1]:

- **Question:** You are to draw a sequence of equidistant telegraph poles along a straight road in a picture showing the landscape seen in a bird's eye view, with the horizon 6 cm from the first pole, and the separation between first and second pole 1 cm (see figure below). Compute where the third pole should be. Extend this to computing the location of the k-th pole. (Hint: Compute the cross ratio of the first two poles to the point at infinity in a 'straight' photograph. Then realize that the cross ratio is a projective invariant.)



- **Answer:** Let us solve this for pole k, numbering the poles from zero. We compute a cross ratio as in Figure 1 for p as pole 0, q as pole 1, r as pole k and s as the pole at infinity. That cross ratio, as a function of k, is $\big((1)(\infty - k)\big)/\big((k-1)(\infty)\big) = 1/(k-1)$. (If you are uncomfortable with $\infty/\infty = 1$, just use the pole numbered googolplex instead.) Since the cross-ratio is a projective invariant, it should be the same in the picture we draw. If pole k is drawn as a distance x from pole 0 in the picture, we should therefore have: $\big((1)(6-x)\big)/\big((x-1)(6)\big) = 1/(k-1)$. It follows that $x = 6k/(k+5)$ is the location, in centimeters, of pole k. The third pole ($k = 2$) should therefore be drawn $5/7$ cm further along the line.

2.6 Circular Points (page 52)

The circular points are unique, but they are points in homogeneous coordinates, using complex coefficients. You are therefore allowed to use complex numbers as the homogeneous scaling factor. So the point $(1, i, 0)^\top$ is equivalent to $(i, -1, 0)^\top$ (multiply by i) and many other forms. Apart from that, the two circular points are indeed distinct and unique.

3 Chapter 3

3.1 What's in a Name (pg 65)

This book refers to projective transformations as *collineations*. Some other books, often based in French research, call them *homographies*. The terms are identical. Other terms used are *projectivities* (again, the same) and *perspectivities* (these are different in that they make a distinction between rays entering the eye from the front and from behind, which is of course what we would actually want).

3.2 Nullspaces and Spans (pg 67)

The (right-) *nullspace* is sometimes called the *kernel* of a transformation. It is expressed as a *span* of some basis vectors. These English terms in linear algebra are almost equivalent to defining products on vectors, for they are linearly related to the vectors you put into them. In 3D, for instance, if you have two vectors $\mathbf{a}$ and $\mathbf{b}$, then make the matrix with these vectors as columns: $[\mathbf{a}, \mathbf{b}]$, and ask for the nullspace of the transpose, then you have:

$$\text{nullspace}[\mathbf{a}, \mathbf{b}]^\top = \text{span}\{\mathbf{a} \times \mathbf{b}\},$$

for you are asking for vectors that are perpendicular to both $\mathbf{a}$ and $\mathbf{b}$, and those are of course a mulptiples of the cross product. This chapter does a lot of similar things, for more vectors, in different dimensions.

Linear algebra has no structural way of defining the nullspace as a product, because it has no algebraic way of representing higher dimensional subspaces as elements of computation. Geometric algebra (or Grassmann-Cayley algebra) does, and in it $\mathbf{a} \wedge \mathbf{b}$ is the quantitative oriented 2-D space elements spanned by $\mathbf{a}$ and $\mathbf{b}$. That construction generalizes to arbitrary dimensions and subspaces. The nullspace of the transpose is a characterization of that subspace by vectors that are perpendicular to the subspace. In GA, that would be explicitly indicated as $\mathbf{a} \times \mathbf{b} = (\mathbf{a} \wedge \mathbf{b})^*$ (the *dual* of the span).

By the way, the *left nullspace* of a matrix A is defined as the set of vectors satisfying $\mathbf{x}^\top A = 0$.

3.3 The Meaning of Plücker Coordinates (pg 70-73)

HZ do not talk about the geometrical meaning of the Plücker coordinates at all, even though that is very straightforward. The 6 Plücker coordinates of a line can be written as two vectors $[\mathbf{a}, \mathbf{m}]$ (or a multiple of this pair), where $\mathbf{a}$ is the *unit direction vector* of the line, and $\mathbf{m}$ its *moment*, characterizing the origin plane in which the line resides, and its distance. See Figure 2. The Plücker condition is then simply the demand that the moment is perpendicular to the direction vector: $\mathbf{a} \cdot \mathbf{m} = 0$.

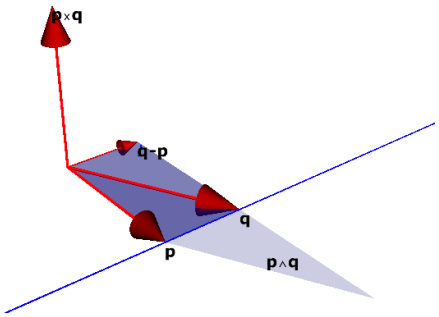


Figure 2: *Plücker coordinates of a line in 3D. (From [1].)*

The line through two points characterized by two vectors $\mathbf{p}$ and $\mathbf{q}$ is simply characterized by the Plücker coordinates:

$$[\mathbf{a}, \mathbf{m}] = [\mathbf{p} - \mathbf{q}, \mathbf{p} \times \mathbf{q}],$$

and this corresponds to the Plücker matrix

$$\begin{bmatrix} 0 & m_3 & -m_2 & a_1 \\ -m_3 & 0 & m_1 & a_2 \\ m_2 & -m_1 & 0 & a_3 \\ -a_1 & -a_2 & -a_3 & 0 \end{bmatrix}$$

When you compare this to the HZ text, you see that the order and signs in their definition of Plücker coordinates in (3.11) is completely garbled, though all elements are there.

3.4 Twisted Cubic (pg 75)

What HZ call a twisted cubic is just a local representation of the simplest truly 3-D curve. Locally developing any trajectory in terms of its time parameter θ gives that representation. I would never view that as the generalization of a conic (that really *is* twisted!), but as the straightforward generalization of a local curve representation.

The parametrization of the same curve does not affect how you interpret it as a point set, so θ can be deformed on all sorts of ways. HZ permit only distortion by projective transformation, which supposedly makes sense in the context of their representation (even though it is a restriction).

3.5 A Typo (pg 84)

There is a rather obvious type 3 lines above Result 3.10: replace Ω_∞ by $\mathsf{Q}^\star_\infty$.

3.6 A Matter of Definition

Equation (3.25) and similar equations: this is not just a way to compute the cosine, it is the only way to *define* it within the non-metric context.

4 Estimation of 2D Projecctive Transformations

4.1 DLT (pg.90)

For an $n \times m$ matrix, acting on $m \times 1$ vectors, all vector components need to go somewhere. This is expressed in a relationship between the dimensionality of the *image* (the somewhere that vectors end up in) and the *kernel (nullspace, the vectors that go to zero)* of the matrix:

$$\dim(\text{im}(A)) + \dim(\ker(A)) = m$$

The dimension of the image is the *rank*. Therefore an 8×9 matrix of rank 8 has a 1-dimensional nullspace.

4.2 Algebraic Distance (pg. 93)

The formula at the bottom is of course not valid in general. $\mathbf{x}_1$ and $\mathbf{x}_2$ *must* be the standard homogeneous 3D representations of two 2D points, with a 1 as third coordinate. Then you can derive the formula simply. But all this context means that it is not necessarily 'more briefly'.

4.3 Geometric and Algebraic Distance (pg.96)

The relationship between algebraic and geometric distance is relatively simple, with only the (unknowable) homogeneous factors w_i needed for correction. But since these factors are not part of reality, the algebraic distance is somewhat unrealistic. You can more or less (not exactly as Leo suggested) imagine it as the distance between two representative points on the homogeneous 3D ray representing 2D points.

4.4 Gold Standard (pg.114)

HZ appear to mix some ideas here, for their convenience, putting a RANSAC initialization inside their Gold Standard algorithm. It would have been nicer to keep things totally pure. But since it is only initialization, Table 4.1 is still usable enough.

4.5 RANSAC (pg.118)

The algorithm given is not quite the original RANSAC, the re-estimation step iii is not original (but often done).

4.6 Pre-emptive RANSAC

The most expensive step of RANSAC is to compare the rest of the data to a model based on the few selected inliers. Nowadays, there is a method that can perform a greedy search on this fit, allowing early abortion. This is called 'pre-emptive RANSAC', and a reference is *Preemptive RANSAC for Live Structure and Motion Estimation*, by David Nistér (2003).

5 Algorithm Evaluation and Error Analysis

5.1 RMS (pg.133)

Without the factor of 1/2, this would be precisely the average distance. We wonder why they put it in.

5.2 X (pg.134)

MZ go into math mode here. The notations M, N, $\mathbf{X}$ are not really specific, although their constant illustration with 2D estimation then causes some confustion. In that illustration, $M = 8$ (the number of true degrees of freedom in $\mathtt{H}$), $N = 2K$ when using K data points to establish correspondence, and $\mathbf{X}$ corresponds to the primed $\mathbf{x}'_i$ in the second image.

5.3 The function f (pg.134)

There was some confusion about the definition of f and its argument $\bar{\mathbf{P}}$. In the case studied, the $\bar{\mathbf{P}}$ contains the (8) *variables* in $\bar{\mathtt{H}}$, but the function f is also determined by $\bar{\mathbf{X}}$ as its *parameters*. One could denote this dependence as $f(\bar{\mathtt{H}}; \bar{\mathbf{X}})$.

5.4 $N\sigma^2$ (pg.134)

The $N\sigma^2$ is not just a 'notation', it is the exact result.

5.5 Unfortunate drawing? (pg.135)

Drawing 5.2 appears to do the linear approximation at a maximum, or a ridge, of the S_M-surface. That makes for a clear picture, but it is actually a location where the linearization is rather bad.

5.6 MLE (pg.135)

The actual MLE estimator should be the point closest to $\mathbf{X}$ on the surface S in the proper metric on $\mathbb{R}^N$. If the surface is locally planar at $\bar{\mathbf{X}}$, this can be approximated by the projection onto the tangent plane at $\bar{\mathbf{X}}$. HZ call this the MLE, but it would have been better to call it 'linearized MLE' or some such term.

5.7 Convergence (pg.138)

It is very strange to base a convergence criterion on (5.7). All this equality would test is if Pythagoras is valid, i.e. if the angle between $\mathbf{X} - \bar{\mathbf{X}}$ and $\hat{\mathbf{X}} - \bar{\mathbf{X}}$ is a right angle. It would be better to make this a proper angle computation, now the result depends on the magnitude of the $\mathbf{X}$s (if one compares different estimators at the same $\mathbf{X}$ that might be OK). Also, all one would test is if the estimator converges to the 'linearized MLE' (see previous remark). The bullets indeed modify the strong claim, but in a manner that makes one doubt its usefulness.

5.8 Typo (pg.145)

Halfway page 145, 'an 8-dimensional surface S_P' should be 'an 8-dimensional surface S_M'.

5.9 Monte Carlo (pg.149)

A lot more might have been said about the Monte Carlo method, which needs to be used properly. *Does anyone have a tutorial reference?*

5.10 Inverse via SVD (pg.592)

The remark below A5.3 seems to say that computing the inverse of an $m \times n$ matrix `A` is expensive via the SVD method if $n << m$. But in that case, one just does the SVD of $A^\top$, which is related in a straightforward manner to that of A. Daniel remarks that both 'taking the inverse' and 'making the SVD' are of the order $O(m^3)$, though Golub and Van Loan give a constant factor of about 20 more for the SVD.

6 Camera Models

6.1 Camera Rotation and Translation (pg.155)

HZ are a bit sloppy when specifying the matrix (6.6). In this formula, the translation is measured by $\tilde{\mathbf{C}}$ (camera center in world coordinates), whereas the `R` matrix is the orientation of the world frame relative to camera coordinates. Had they been consistent and used the rotation matrix to denote the rotation of the camera frame relative to the world frame, the matrix would have been $\begin{bmatrix} \mathtt{R} & \tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix}$.

6.2 Vertigo Effect (pg.166)

Apparently, some of you watch the tour de France (why?), where this effect appears when a motorcycle camera zooms in on a cyclist while the motor cycle

slows down.

6.3 World Origin in Affine Cameras (pg.167)

It puzzled us that the world origin should play such an important role in an affine camera. The world origin has no absolute meaning. How do we resolve this?

6.4 Weak Perspective (pg.170)

Who can give the derivation that connects the geometrical interpretation of Fig.6.8 with the weak perspective formula (6.25)?

6.5 d and C

In the camera models, a confusion may arise between **d** and **C** in infinite and finite cameras. The relationship in given in Table 6.1, and the distinction would really disappear if we would treat **d** as 'the camera center infinity'.

7 Computation of the Camera Matrix

7.1 Critical Configurations (pg.179-180)

Condition (i) is surprising, and not explained. Condition (ii) is somewhat hard to parse, but seems to mean: the points may lie in a plane (which is obviously degenerate), but even lifting some of them off the plane may retain degeneracy if those still all lie in a single straight line through the camera center.

7.2 Restricted Camera Estimation (pg.185)

Daniel remarked that the principal point may be 50 pixels off for 1000×1000, and that this depends on the lens. Principal points may need to be re-estimated often, especially when using zoom lenses or automatic motion stabilization cameras.

7.3 Reduces Measurement Matrix (pg.186)

Using $\hat{A}$ reduces the complexity for LM (the linear method is still as expensive). Olaf used to use it, but for RANSAC he found that *and here I cannot decypher my notes, Olaf please send me your remark again.*

7.4 Radial Distortion (pg.191)

If the principal point is estimated well, the distortion should be an even function of r (the distance to the principal point). Other authors (like Zhang) use the model $L(r) = 1 + \kappa_1 r^2 + \kappa_2 r^4 + \cdots$. Is this a typo in HZ?

8 More Single View Geometry

8.1 A Star of Planes (pg.197)

How can a two-parameter family have a three-dimensional basis? We don't know.

8.2 An Unlikely House (pg.203)

You should redraw the 3D house in Fig.8.5.

8.3 Short Calculation (pg.203 line -4)

Olaf's short calculation used that the skew was zero, and he wondered whether that generalizes. Who knows?

8.4 Rotation Angle from Matrix (pag.204)

A simpler way to get a rotation angle from a matrix is through: $\phi = (\text{acos}(\text{trace}(R)) - 1)/2$. The sign of the angle is determined by the chosen orientation for the rotation axis.

8.5 Calibration (pg.211)

Zhang gives a more down-to-earth derivation of (8.12) and then relates it to the absolute conic (possibly because H or Z were reviewers of his paper). We will see this later when Daniel treats Zhang.

8.6 Vanishing Points (pg.213)

Olaf points out that HZ could avoid treating vanishing points as a special case, within the projective context. But at least they finally give some intuition on them.

9 Epipolar Geometry

9.1 Derivation of Lemma 9.11

The step '$[\mathbf{a}]_\times(k\tilde{\mathtt{A}} - \mathtt{A}) = 0$ and so $(k\tilde{\mathtt{A}} - \mathtt{A}) = \mathbf{a}\mathbf{v}^\top$ for some $\mathbf{v}$' is not completely trivial, but if you apply both sides to a vector $\mathbf{y}$ you see that this becomes equivalent to stating that $\mathbf{a} \times \mathbf{a} = 0$.

9.2 The Essential Matrix (pg.257)

As Olaf said, to roboticists the essential matrix is more essential than the more fundamental fundamental matrix, because we often have one calibrated camera with which we have to understand the world. The derivation of the essential matrix can de done fairly directly once you realize that $\hat{\mathbf{x}}^\top [\mathbf{t}]_\times R\mathbf{x} = 0$ can be rewritten to $\hat{\mathbf{x}} \cdot (\mathbf{t} \times R\mathbf{x}) = \det(\hat{\mathbf{x}}\ \mathbf{t}\ R\mathbf{x}) = 0$ and therefore expresses the coplanarity of $\mathbf{x}'$, $\mathbf{t}$ and $R\mathbf{x}$ (all vectors that can be considered to reside in the primed coordinate system). That is basically Fig.9.1.a, where the calibration allows us to consider $\mathbf{t}$, $\mathbf{x}$ and $\hat{\mathbf{x}}$ as absolute directions in space.

11 Computation of the Fundamental Matrix

11.1

The Frobenius norm between matrices is just the square root of the sum of squared differences between corresponding matrix elements. It is the same as the Euclidean distance between the matrices written as vectors (as we often do in solving equations). It is therefore an algebraic distance, not necessarily meaningful to the geometric interpretation of the matrix. The Frobenius norm of a matrix can be rewritten in terms of the sum of the squares of the singular values, which helps in its interpretation.

11.2 The minimum case (pg.281)

The general solution would be $\alpha F_1 + \beta F_2$, but as always the scale of F is immaterial so we can take the affine combination $\alpha F_1 + (1-\alpha)\beta F_2$ and not lose anything.

11.3 Algebraic Minimization

Gijs noted an issue: if you want to apply Levenberg-Marquardt, you need a Jacobian. What is the Jacobian in this section? it is also not specified in Algorithm 11.2.

11.4 Envelope of Epipolar lines (pg.300)

The step from (11.12) to (11.13) had me puzzled, but it is simply a matter of applying the definition of the squiggled $\mathbf{m}$s to rewrite them to the unsquiggled ones, and similar for Σ according to the definition just below (11.11).

15 The Trifocal Tensor

15.1 Tensor Notation

Finally, on page 376, HZ make a distinction between points and hyperplanes, as column or row vectors. This is very convenient, as it makes tranformations universal. For instance, performing a homography H on a point $\mathbf{x}$ is in matrix form $H\mathbf{x}$, whereas the same homography applied to a hyperplane (line in 2D) Π is in matrixform $H^\top\Pi$. so you, and your software, need to know whether the vectors were intended to represent points or lines. This is extra administration, and extra code.

In tensor notation, this is completely unnecessary. A point is characterized on a frame $\{\mathbf{e}_i\}$ as $\mathbf{x} = x^i\mathbf{e}_i$, and a hyperplane on the reciprocal frame as $\Pi = \pi_i\mathbf{e}^i$. The transformation of either is according the formula $x^i \mapsto x^j = H^j_i x^i$ or $\pi_j \mapsto \pi_i = H^j_i \pi_j$. The tensor H^j_i therefore transforms either. No extra code, no reason for bugs causing inconsistency (which are moreover hard to trace if you have not even had the discipline to pick your data structures carefully, for coordinate representations do not contain enough information to tell you what is going on geometrically).

Tensors can be extended to transform arbitrary linear elements such as 3D lines when using Plücker coordinates, where they should incorporate the natural transformation law $H(\Lambda) = H(x \wedge y) \equiv H(x) \wedge H(y)$, where $\wedge$ denotes the join operation for linear elements (in this case making a line Λ out of two points x and y). You need the slightly more general framework of Grassmann-Cayley algebra to represent such elements naturally. If you need metric properties, upgrade to the next level, which is geometric algebra.

16 Appenddix 6: Iterative Estimation Methods

16.1 Newton's method (pg.598)

The dual explanation of Newton's method is a bit confusing. The first time, they use the model f to define the error and derive the corresponding least squares problem. The second explanation is intended to be more general, starting from an error cost function g (which they relate to the f just before). We will later see how to manipulate such cost functions, so this is a useful abstraction.

16.2 Gradient Descent (pg.599)

It is a bit surprising that the *conjugated gradient* method is not mentioned. It can avoid the zigzagging, see figure 3.

16.3 Sparseness (pg.602-608)

Daniel treated this sparsely, it is a rather specialized subject. How to split one's parameter matrix depends very much on the application. Olaf's robot

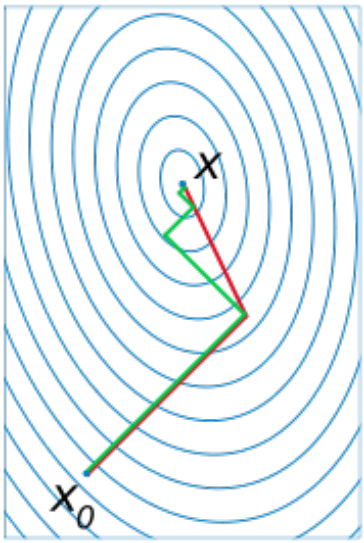


Figure 3: *(From http://en.wikipedia.org/wiki/Conjugate_gradient_method) The red line is iterative conjugated gradients, the green line is iterative gradient descent.*

localization does it for image data not visible in all frames; Daniel's skeleton estimation has some hierarchical sparseness in teh kinematic chain parameters. But it is good to know that fairly structural methods to treat the sparseness are available when we need them.

16.4 Robust Cost Functions (pg 616)

The inpendence of the outliers for a cost function that becomes linear (as in L1, Huber and pseudo-Huber) may require some explanation. If you have a 1D data set of localized points x_i, and want to find the a location that minimizes the squared distance with all points, then this is the average, or mean, location. If you want to minimize the absolute distance, then this is done by the middle point (if the total number of points is odd) or by any location between the two 'middle points' (if you have an even number). In the latter case, any location between those will do: if you shift a bit, what you lose on the left you gain on the right. The location hence depends not on the precise location of the data points, whether outliers or not, just on the location of the middle points. Therefore a linear cost function is insensitive to outliers. More precisely: their location does not matter at all, and whether they lie to the left or the right of the ensemeble only influences the result a bit – but even then, the optimum only shifts to the next 'middle' point(s), and is still determined by the core data.

16.5 RANSAC/Huber (pg.620)

We wondered about the combination of RANSAC and Huber. Nowadays, people use RANSAC to focus on a data set in which many outliers are already removed, say to go from 30% outliers to 10%. Then one can use a Huber-like cost function for the iterative optimization. But one could mix the methods using Huber within RANSAC, or RANSAC after the Huber step. We do not know whether this might improve results or convergence speed. Olaf found the paper by Thor and Muray useful [3].

16.6 Sinc (pg.625)

The sinc in line 3 is not a typo, the definition is $\mathrm{sinc}(x) = \sin(x)/x$. But Leo suspects that the proper normalization to get a unit vector would necessitate a factor of $1/2$ in front of the sinc.

17 The Zhang paper

17.1 Availability

Zhang's method, including corner detection for the calibration patterns, was implemented by Bouguet, as a toolbox in Matlab and later in OpenCV. It includes non-uniform radial distortion, though Daniel tells us that those are mostly zero for normal cameras and lenses.

17.2 General remarks

In Daniel's experience, it is important to estimate different internal camera parameters for f_x and f_y. Allowing this leeway gives better 3D reconstruction. The camera center can be considerably different from the middle of the image. Anything more than about 10 pixels has a significant effect. Zhang claims that a slant of about 45 degrees gives the best calibration results, but this is in simulation; when preceded by a corner detection, a more straight-on view would probably be optimal. Olaf remarks that fish-eye lenses may not be describable with a single viewpoint.

17.3 Orthogonalization

Zhang computes $\mathbf{r}_1$ and $\mathbf{r}_2$, then from them $\mathbf{r}_3$, and makes an orthonormal set from those three using the SVD. A simpler, more insighful, and cheaper method to achieve the exactly same result is to 'spread the legs' of $\mathbf{r}_1$ and $\mathbf{r}_2$ geometrically to make them right-angled (Daniel's modification), and compute $\mathbf{r}_3$ from the result.

References

[1] Leo Dorst, Daniel Fontijne, Stephen Mann, *Geometric Algebra for Computer Science*, Morgan Kaufmann, 2007.

[2] Jorge Stolfi, *Oriented Projective Geometry*, Academic Press, 1991.

[3] P.H.S.Torr and D.W.Murray, *The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix*, `http://www.robots.ox.ac.uk/ActiveVision/Publications/torr_murray_ijcv1997/torr_murray_ijcv1997.html`.