Direct Linear Transformation (DLT)

The method consists of two stages. First, it estimates a projection matrix relating the object space volume to image space coordinate. Then, from the entries of the projection matrix, camera parameters are extracted based on geometric properties with algebraic methods.

# Projection matrix

The projection matrix maps a 3-D object point in homogeneous frame to projected image plane. In DLT, it is a 3x4 matrix satisfying the relation written in Eq. (1).

(1)

With an arbitrary scale factor, , from collinearity equation, image coordinate is written as in (2)

(2)

The estimated projection matrix has 11 degree of freedom so that it requires the minimum of 6 2D-3D points correspondences, and a least square approach is engaged with all control points in calibration object. We rewrite the relation of correspondence (1) using a linear homogeneous system as in Eq. (3).

(3)

With

and

***N*** is the number of control points. From Eq. (3), can be obtained up to scale using Singular Value Decomposition (SVD) technique as below.

Because the norm of solution in Eq. (3) is arbitrary, the solution could be of unit vector in least square sense to minimize Eq. (3).

**,**

(4)

With a constraint, . Then, a solution can be found minimizing Eq. (5), a Lagrange equation.

(5)

By taking derivative with respect to , and by equating the equation to 0, we have

(6)

Eq. (6) is an eigen system, with and serving as eigenvalue and eigenvector respectively. From the orthonormal property of eigenvector, Eq. (5) becomes

(7)

This minimizes with the smallest eigenvalue of . From the properties of singular value decomposition, the solution is the right singular vector, of, associated to zero singular value.

DLT::\_Projection() contains the implementation of the aforementioned methods.

Note: The coordinate of image and object points to DLT method are normalized in order to avoid numerical accuracy issue, and the technique is discussed in the later part of the document.

# Parameter Decomposition

After having entries of projection matrix using the least square sense, we introduce a few variables from the projection matrix as in Eq. (8).

(8)

Because entries of are scaled 3rd row of rotation matrix, and from the orthonormality, the scale factor, , is discovered up to sign as in Eq. (9).

(9)

*After having the resolved scale applied to least square solution*, principal point , is recovered by taking the dot product of with and .

(10)

Similarly,

(11)

And, principal point is readily computed from Eq. (12)

(12)

Having all intrinsic camera parameters and and are discovered, the estimation of and follows up to sign, .

(13)

The entries of two rows of rotation matrix are recovered up to sign from as in Eq. (14).

(14)

The method described in two section is one of variant in classical DLT, and should work well if the projection matrix is estimated accurately. This leads two topics to discuss; 1. Stable projection matrix estimation 2. Forcing orthonormality from the numerical solution in Eq. (14).

DLT::\_Decompose() contains the implementation of this section.

# Resolving sign

The sign of 3rd row of projection matrix is determined by the origin of object frame with respect to that of camera; The collinearity equation should be positive if the object frame origin is observable, otherwise negative. While the value of focal lengths and principal point are assumed to be positive, the sign of remaining components in Eq. (13, 14) should be exercised. The correct sign minimizes the cost Eq. (1).

DLT::\_SwitchSign() contains the implementation of this section.

# Orthonormal Rotation Matrix

All entries of the rotation matrix are estimated, but rotation matrix consisting of the least square solution does not satisfy orthonormality. This section describes how orthonormal rotation matrix close to the numerical solution from Eq. (9, 14) is computed in the code implementation. The content of this section is largely form Zhang [2].

Let be the numerical solution obtained in ‘Parameter decomposition’ section. The rotation matrix, being sought is with property

subject to

(15)

Eq. (16) describes the property of Frobenius norm given a real matrix, .

(16)

Using Eq. (16), a part of Eq. (15) is rewritten as in

(17)

Therefore, finding minimizing Frobenius norm in Eq. (15) becomes the problem of maximizing in Eq. (17).

Let SVD of be , and rewrite the cost to maximize

(18)

Note the property of trace of matrix applied in 2nd and 3rd term in (18), . Also note that

,

(19)

therefore  *is orthonormal, and no entries in exceeds in magnitude*. From the property of trace of (singular value) diagonal matrix, , with orthonormality, we have

(20)

is *i*-th singular value of and is not dependent to . Eq. (20) is maximized if or . This leads to

(21)

Eq. (21) is true with

(22)

DLT::\_Decompose() contains the implementation of this section.

# Data Normalization

As shown in the parameter decomposition section, the estimation of an accurate projection matrix is critical in DLT. In practice, having many control points, the solution in least square sense becomes less theoretical, but it is an approximate. As a result, non-zero singular value becomes small or we have multiple solution of which singular value approaches to zero.

The data normalization algorithm implemented is based on Chojnacki[3]. The algorithm computes similarity transformation so that input data set is translated and scaled as below.

Given sets of input data to DLT algorithm, let the centroid of image and corresponding object points in homogenous frame be , respectively.

(23)

Where and . The data sets are shifted to the centroids, then scaled as in Eq. (24).

(24)

With a similarity transformation, and , the process makes the root mean square distance of normalized data, and , to its origin, , where *N* is the dimension of data set.

(25)

Each input data set is processed using Eq. (25) and a least square approach follows in order to find projection matrix in normalized data sets. Finally retrieve the original projection matrix by following formula.

(26)

DLT::\_Normalize() contains the implementation of this section.

## References

[1] Trucco and Verri, “Introductory Techniques for 3-D Computer Vision”, Prentice Hall, ISBN 0-13-261108-2

[2] Zhang, “A Flexible New Technique for Camera Calibration”, pp 1330-1334, IEEE PAMI Vol 22, Issue 11, Nov 2000

[3] Chojnacki and Brooks, “Revisiting Hartley’s Normalized Eight-Point Algorithm”, pp 1172-1177, IEEE PAMI Vol 25, Issue 9, Sept. 2003