24

Projective Reconstruction and Its Application in Object Recognition for Robot Vision System

Ferenc Tél and Béla Lantos
Budapest University of Technology and Economics
Hungary

1. Introduction

More and more applications (path planning of a robot, collision avoidance methods) require 3D description of the surround world. This chapter describes a 3D projective reconstruction method and its application in an object recognition algorithm.
The described system uses 2D (color or grayscale) images about the scene taken by uncalibrated cameras, tries to localize known object(s) and determine the (relative) position and orientation between them. The scene reconstruction algorithm uses simple 2D geometric entities (points, lines) produced by a low-level feature detector as the images of the 3D vertices and edges of the objects. The features are matched across views (Tél & Tóth, 2000). During the projective reconstruction the 3D description is recovered. The developed system uses uncalibrated cameras, therefore only projective 3D structure can be detected defined up to a collineation. Using the Euclidean information about a known set of predefined objects stored in database and the results of the recognition algorithm, the description could be updated to a metric one.

Projective reconstruction methods

There are many known solutions to the projective reconstruction problem. Most of the developed methods use point features (e.g. vertices), but there are extensions to use higher order features, such as lines and curves (Kaminski & Shashua, 2004). The existing methods can be separated into three main groups. The *view tensors* describe the algebraic relationships amongst coordinates of features in multiple images that must be satisfied in order to represent the same spatial feature in 3D scene (Faugeras & Mourrain, 1995). These methods estimate fundamental matrix from two views (Armangué et al., 2001) or trifocal tensor from three views (Torr & Zisserman, 1997). The *factorization based methods* use the fact that collecting the weighted homogeneous (point) projection vectors into a large matrix (measurement matrix), the rank must be four, because it is a product of two rank four matrices. An iterative solution to solve this problem can be found in (Han & Kanade 2000). In *bundle adjustment methods* the reprojection errors between original image feature locations and an estimated projection of spatial feature locations are minimized. The solution for the problem can be found applying e.g. nonlinear least squares algorithm (Levenberg-Marquardt).

Object recognition methods

The aim of object recognition methods is to recognize objects in the scene from a *known* set of objects, hence some *a-priori* information is required about the objects. These types of

Open Access Database www.i-techonline.com

Source: Computer Vision, Book edited by: Xiong Zhihui,
ISBN 978-953-7619-21-3, pp. 538, November 2008, I-Tech, Vienna, Austria

methods are called *model based* object recognition methods, where predefined object databases are used to store the required information about the objects. There are many classification types of the applied (and stored) object models, such as object-centered or viewer-centered models, physical or geometrical models, rigid or deformable models, etc. The dimensionality of the used information is also changed in different recognition systems, there are 2D only, mixed and 3D only systems. The developed algorithms are usually evaluated against set of different criteria, such as search complexity, discriminative power and robustness. The *appearance based methods* use 2D images as object representations. Using multiple views, the stored information can be reduced to a minimal set. Here the intensity distribution of the images is used as the basis of the comparison of the similarity of the projected intensity image among views. The two different strategies are global ones, e.g. eigenface (Belhumeur et al., 1997) or local approach, where local properties of the images (neighborhood of edges or corner points) are used to improve the discriminative power, e.g. GLOH (Mikolajczyk & Schmid, 2005)).

The *aspect graph methods* use the changes in the projected geometry of the objects and group views bounded by transitions of the geometry (Schiffenbauer, 2001). The information reduction is based on the determination of general views, which are equivalent with each other.

The *indexing based methods* use those properties of the data that are invariant against a selected group of transformations. In this case the transformation describes the relationship between the object data as stored in the database and scene information, therefore the transformations could be rigid (translation and rotation), similarity (rigid and scaling), etc., up to the most general projective one (collineation). The most widely used methods are based on the geometric hashing (Wolfson & Rigoutsos, 1997). In this case subsets of features (points) are selected that can be used to form a basis and define local coordinate system with that basis. Calculating the coordinates of all of the remaining features in this coordinate system and quantizing the calculated coordinates a hash table is constructed. During the query a similar method is applied and vote is generated into the respecting entry of the hash table.

Euclidean update methods

The last step of the reconstruction is (if the robot control application requires) the update of the reconstructed data from projective to a metric one. There are several algorithms that address this issue. One group of applications uses known a-priori information to recover metric information. In (Boufama et al., 1993) e.g. the coordinates of known points, points laying on the plane of the given (reference) frame, known alignment of points on vertical or horizontal line and known distance between points are used to involve metrical information into the reconstruction. In (Faugeras, 1995) an update sequence is described, that converts the reconstruction from projective to affine, then from affine to Euclidean. The proposed a-priori information is either the known motion of the camera, parallelity of lines (for affine) or angle between lines (for Euclidean) reconstruction.

The other type of methods uses the hypothesis of fixed (but unknown) intrinsic camera parameters. These algorithms are known camera self-calibration methods. This yields the intrinsic parameters of the cameras using only imaging information. (Hartley, 1993) supposes that the cameras have common calibration matrices and uses nonlinear minimizations to calculate camera matrices. A huge nonlinear minimization is achieved to get the final description.

The chapter is divided in sections as follows. Section 2 gives an overview of the most important methods of projective reconstruction. The main part of the chapter is section 3 dealing with object recognition based on a new indexing method. Section 4 presents a method of Euclidean reconstruction assuming uncalibrated cameras for robot applications if the goal is to find the relative position and orientation between the gripper and the recognized object. Section 6 contains the conclusions and some directions of future developments. Section 7 is the Appendix summarizing the basic results of projective geometry and notations used in the chapter.

2. Projective reconstruction

The developed system uses two types of reconstruction algorithms, the first uses point features only and the other uses point and line features together.

2.1 Cost function for points

Using the pinhole camera model the projection equation for points can be written into linear form $\rho_{ij}\mathbf{q}_{ij} = \mathbf{P}_i\mathbf{Q}_j$. In this case the scale factor ρ_{ij} denotes the projective depth of the given point. If there are m cameras and n_P points in the scene, then the number of projected image points (and scale factors) are $m \times n_P$. But only $m + n_P$ are independent amongst them, therefore the projective depths should be decomposed into camera dependent and feature dependent parts. The decomposition equation can be written as a product of two other quantities: $\rho_{ij} = \pi_i\gamma_j$. Using this decomposition, the projection of a point is described by $\pi_i\gamma_j\mathbf{q}_{ij} = \mathbf{P}_i\mathbf{Q}_j$. This decomposition has some advantages: i) the system is described with the minimum number of parameters, therefore the parameterization is consistent. ii) the number of unknowns is greatly reduced. E.g. $m = 3, n_P = 40 \rightarrow N(\pi_i, \gamma_j) \leq 43 << N(\rho_{ij}) = 120$. If the ρ_{ij} projection depths were known, the joint projection matrices $\mathbf{P}_i$ and the projective shape $\mathbf{Q}_j$ could be determined by using a rank 4 decomposition method, this is the base of the factorization methods.

In order to minimize a physically meaningful quantity, the weighted reprojection error used in the cost function has the form

$$E_P(\cdot) = \sum_{i=1}^{m}\sum_{j=1}^{n_P} \omega_{ij}^2 \left\| \pi_i\gamma_j\mathbf{q}_{ij} - \mathbf{P}_i\mathbf{Q}_j \right\|^2 \tag{1}$$

where the unknowns are $\pi_i, \gamma_j, \mathbf{P}_i, \mathbf{Q}_j$.

2.2 Cost function for points and lines

At first sight it seems a natural choice to extend the decomposition algorithm to lines simply writing the line projection equations into similar form as in the points-only case using the line projection matrix $\mathbf{G}_i$, see (A7) in Appendix:

$$E_L(\cdot) = \sum_{i=1}^{m}\sum_{j=1}^{n_L} \omega_{ij}^2 \left\| \pi_i\lambda_j\mathbf{l}_{ij} - \mathbf{G}_i\mathbf{\Lambda}_j \right\|^2$$

But unfortunately i) the projective depth could not directly be interpreted for lines, ii) the mapping between elements of the point and the line projection matrices is a non-linear function, iii) there exists no distance metric that can easily (linearly) be expressed with the terms of 2D line.
Therefore the original error function (1) was modified. The calculation of projective depths was eliminated using a cross product instead of difference, namely $\mathbf{q}_{ij} \sim \mathbf{P}_i\mathbf{Q}_j \rightarrow \mathbf{q}_{ij} \times (\mathbf{P}_i\mathbf{Q}_j) = 0$. This error is an algebraic distance, it describes the incidence relation between the true (2D feature point) and the projected point. For lines, similar error metric (geometric configurations) was defined:

- The incidence relation of 2D line feature and a projected 3D point is $\mathbf{l}_{ik}\mathbf{P}_i\mathbf{Q}_k(\mathbf{\Lambda},t) = 0$, where $\mathbf{Q}_k(\mathbf{\Lambda},t)$ is the t'th point on the $\mathbf{\Lambda}$ 3D line in Plücker representation (A3). The points can be extracted from Plücker matrix using SVD, see (A5) and (A25). This form can be used during the calculation of $\mathbf{P}$ matrices (resection phase).
- The identity relation of the 2D line feature and a projected 3D line is $\mathbf{l}_{ik} \times (\mathbf{G}_i\mathbf{\Lambda}_k) = 0$. This form can be used during the calculation of $\mathbf{\Lambda}$ vectors (intersection phase).
- The containment relation of 3D line and a plane. The plane can be determined as a backprojected 2D line: $\mathbf{S}_{ik} = \mathbf{P}_i^T\mathbf{l}_{ik}$. The line $\mathbf{\Lambda}_k$ lies on the plane if $\mathbf{U}(\mathbf{S}_{ik})\mathbf{\Lambda}_k = 0$, where $\mathbf{U}(\mathbf{S}_{ik})$ is defined by (A10) in Appendix. This form can be used during the calculation of $\mathbf{\Lambda}$ vectors (intersection phase).

2.3 Minimization of the cost functions

It can be seen, that the cost functions $E_P(\cdot)$ and $E_L(\cdot)$ are nonlinear in the unknowns and their minimization is similar. A possible solution could be the use of the Levenberg-Marquardt method and general initial values to directly minimize this cost function. But fortunately the parameters to be estimated can be separated into different groups, because they are "independent" from each other (e.g. 3D features are independent from each other, because they depend only on the objects in the scene and they are not influenced by the projections). This is the well-known resection-intersection method that holds every group of parameters fixed, except those, that are currently minimized. Therefore the minimization of $E_P(\cdot)$ can be achieved in repeated steps. After every iteration the revaluation of the ω_{ij} weighting factors are achieved and the actual value of the cost function is calculated. If the cost is less than a desired threshold (or maximum allowed number of iterations is reached), the algorithm terminates. The estimation of the given entity can be calculated by making the derivative of $E_P(\cdot)$ by the respecting entity to zero and the solution can be found in closed form for each of the features, see (Tél & Lantos, 2007) for details.
For the more general mixed case the detailed calculations are as follows. The error function for the *intersection phase* is

$$E_I(\cdot) = \sum_{i=1}^{m}\sum_{j=1}^{n_P} \omega_{Q,ij}^2 \left\| \mathbf{q}_{ij} \times (\mathbf{P}_i\mathbf{Q}_j) \right\|^2 + \sum_{i=1}^{m}\sum_{k=1}^{n_L} \omega_{\Lambda,ik}^2 \left\| f(\mathbf{P}_i, \mathbf{l}_{ik}) \right\|^2$$

where $f(\mathbf{P}_i, \mathbf{l}_{ik}) = \mathbf{l}_{ik} \times (\mathbf{G}_i\mathbf{\Lambda}_k)$ or $f(\mathbf{P}_i, \mathbf{l}_{ik}) = \mathbf{U}(\mathbf{S}_{ik})\mathbf{\Lambda}_k$.
During this phase, the $\mathbf{P}_i$ (therefore the $\mathbf{G}_i$) projection matrices are held fixed. After some manipulation the $E_I(\cdot)$ can be written into the following form:

$$E_I(\cdot) = \sum_{j=1}^{n_P}\sum_{i=1}^{m} \omega_{Q,ij}^2 \mathbf{Q}_j^T \mathbf{A}_{I,ij}^T \mathbf{A}_{I,ij} \mathbf{Q}_j + \sum_{k=1}^{n_L}\sum_{i=1}^{m} \omega_{\Lambda,ik}^2 \mathbf{\Lambda}_k^T \mathbf{B}_{I,ik}^T \mathbf{B}_{I,ik} \mathbf{\Lambda}_k$$

where $\mathbf{A}_{I,ij} = [\mathbf{q}_{ij}]_\times \mathbf{P}_i$ and $\mathbf{B}_{I,ik} = [\mathbf{l}_{ik}]_\times \mathbf{G}_i$ or $\mathbf{B}_{I,ik} = \mathbf{U}(\mathbf{S}_{ik})$. The estimation for the *j*'th feature can be calculated by making the derivative of $E_I(\cdot)$ by $\mathbf{Q}_j$ and $\mathbf{\Lambda}_k$ to zero, respectively. After the differentiation the solution for each $\mathbf{Q}_i$ and $\mathbf{\Lambda}_k$ can be found in closed form. During the calculation of $\mathbf{Q}_j$ an additional constraint must be introduced, in order to eliminate trivial all zero case. The solution of the problem for $\mathbf{Q}_j$ is the normalized eigenvector corresponding to the smallest eigenvalue of the matrix $\mathbf{C}_{R,ij} = \sum_{i=1}^{m} \omega_{Q,ij}^2 \mathbf{A}_{I,ij}^T \mathbf{A}_{I,ij}$

During the calculation of $\mathbf{\Lambda}_k$ lines, two additional constraints must be fulfilled. The first one is the elimination of the trivial (all zero) case, the second one is the Plücker constraint for vector $\mathbf{\Lambda}_k$, see (A3). The measurement error part is similar to the point-case but here the matrix is $\mathbf{D}_{I,ik} = \sum_{i=1}^{m} \omega_{\Lambda,ik}^2 \mathbf{B}_{I,ik}^T \mathbf{B}_{I,ik}$. The error function with the constraint can be written into the matrix equation $\mathbf{\Lambda}_k^T (\mathbf{D}_{I,ik} + \alpha \mathbf{\Delta}) \mathbf{\Lambda}_k = 0$. Taking the derivative by $\mathbf{\Lambda}_k$ and rearranging the terms yields $(\alpha := -\alpha)$ $\mathbf{D}_{I,ik} \mathbf{\Lambda}_k = \alpha \mathbf{\Delta} \mathbf{\Lambda}_k$. The matrix $\mathbf{\Delta}$ in (A4) is invertible and $\mathbf{\Delta}^{-1} = \mathbf{\Delta}$, therefore the (approximate) solution of the problem for $\mathbf{\Lambda}_k$ is the vector corresponding to the smallest singular value of the matrix $\mathbf{D}_\Delta = \mathbf{\Delta} \mathbf{D}_{I,ik}$.

The error function for the *resection phase* is

$$E_R(\cdot) = \sum_{i=1}^{m}\sum_{j=1}^{n_P} \omega_{Q,ij}^2 \left\| \mathbf{q}_{ij} \times (\mathbf{P}_i \mathbf{Q}_j) \right\|^2 + \sum_{i=1}^{m}\sum_{k=1}^{n_L} \omega_{\Lambda,ik}^2 (\mathbf{l}_{ik} \mathbf{P}_i \mathbf{Q}_k(\Lambda, t))^2$$

During this phase the $\mathbf{Q}_\mathrm{j}$ and $\mathbf{\Lambda}_\mathrm{j}$ entries are held fixed. Again the cameras are independent from each other. After some manipulation $E_R(\cdot)$ can be rewritten into the form

$$E_R(\cdot) = \sum_{i=1}^{m} \mathbf{p}_i^T \left(\sum_{i=1}^{n_P} \omega_{Q,ij}^2 \mathbf{A}_{R,ij}^T \mathbf{A}_{R,ij} + \sum_{k=1}^{n_L} \omega_{\Lambda,ik}^2 \mathbf{g}(t)_{R,ik} \mathbf{g}(t)_{R,ik}^T \right) \mathbf{p}_i$$

where $\mathbf{A}_{R,ij} = \begin{pmatrix} \mathbf{0}_4^T & -w_{ij}\mathbf{Q}_{ij}^T & v_{ij}\mathbf{Q}_{ij}^T \\ w_{ij}\mathbf{Q}_{ij}^T & \mathbf{0}_4^T & -u_{ij}\mathbf{Q}_{ij}^T \\ -v_{ij}\mathbf{Q}_{ij}^T & u_{ij}\mathbf{Q}_{ij}^T & \mathbf{0}_4^T \end{pmatrix}$ and $\mathbf{g}_{R,ij} = \left(a\mathbf{Q}_j^T(L,t) \quad b\mathbf{Q}_j^T(L,t) \quad c\mathbf{Q}_j^T(L,t) \right)^T$.

The estimation for the *i*'th camera can be calculated by making the derivative of $E_R(\cdot)$ by $\mathbf{P}_i$ to zero. Note, that in this case the error function contains only the "point-form" $\mathbf{P}$ of the projection matrices. An additional constraint must be introduced, in order to eliminate trivial $\mathbf{p} = \mathbf{0}$ case. The solution of the problem is the normalized eigenvector corresponding to the smallest eigenvalue of the matrix

$$\mathbf{C}_{R,i} = \sum_{i=1}^{n_P} \omega_{Q,ij}^2 \mathbf{A}_{R,ij}^T \mathbf{A}_{R,ij} + \sum_{k=1}^{n_L} \omega_{\Lambda,ik}^2 \mathbf{g}(t)_{R,ik} \mathbf{g}(t)_{R,ik}^T$$

2.4 Initialization of the entities

The parameters of the cost function are estimated using an iterative method, therefore an initial estimation for its values is required. The developed initialization algorithm:

1. Choosing a subset (pair) of views and subset of points that can be seen on all of the selected images (note: the developed algorithm chooses the views that have the largest number of point correspondences). Using these points a rank 4 factorization method is achieved. This gives initial estimation for the given projection matrices and for selected points.
2. Calculate the projection matrix of a new (not yet processed) view using the points detected on that view and have the spatial coordinates already determined. This can be achieved in closed form using SVD.
3. Calculate the spatial coordinates of the not-yet initialized points, that have projection on the images with determined projection matrix, by using triangulation-like method (Hartley & Sturm, 1997). This means the determination of a point which has minimal distance from the rays connecting the image points and the camera focal points in least squares sense. The solution can be found using SVD.
4. In order to initialize the line features, the algorithm uses the fact that $\mathbf{M}_{ij} = \mathbf{P}_i^T \mathbf{l}_{ij}$ yields a plane that goes through the optical center of the camera and the projected image of the line. Theoretically these planes intersect in the spatial line. Taking more than two views, the solution can be found using SVD. The matrix $\mathbf{A} = \begin{pmatrix} \mathbf{M}_{ij} & \cdots & \mathbf{M}_{mj} \end{pmatrix}^T$ is a rank 2 matrix, therefore the two left null vectors yield two points whose join yields the desired line equation.

The algorithm repeats steps 2 and 3 until all of the projection matrices are calculated.

2.5 Minimization remarks

The two developed algorithms have some common properties.

- Handling of missing data (features having no projection on the given view) during the minimization is simple, the algorithms skip those entries in the error function that do not have valid $\mathbf{q}_{ij}$, $\mathbf{l}_{ik}$ respectively.
- In order to eliminate the effect of the outliers (caused by badly matched feature projections), the camera matrices are estimated only from some subsets of the features in each iteration cycle. These features are selected in a random way and the projection matrix yielding the smallest reprojection error is used in the further steps. The ω_{ij} weights can be used to make the algorithm more robust, e.g. decrease the influence of features with larger error.

3. Object recognition

The developed object recognition method uses permutation and projective invariant based indexing to recognize known object(s) in the scene. A verification step is achieved to finalize the results.

3.1 Invariants

During the recognition process two sets of entities are used. The first one is the feature sets of the object as stored in the object database. The second one is the features of the recovered

scene. Some elements (a subset) represent the same entity in different context (e.g. two representations of the geometric primitives in different coordinate systems). In order to determine the pairing of the two representations of the same entities the process requires the usage of those properties which are not changing (invariant) between representations.
Formally this can be written into the following form. Let $\mathbf{T} \in T$ denote the (linear) transformation between representations and G denote the geometric structure that describes the configuration. The number of functionally independent invariants can be calculated as

$$N_I = \dim(G) - \dim(T) + \dim(T_G), \tag{2}$$

where T_G denotes the isotropy subgroup (if exists), that leaves G unaffected using $\mathbf{T}$ and $\dim(\cdot)$ denotes the dimension of the given entity.
In case of projective invariants the relation between the two representations (Euclidean object database vs. output of the projective reconstruction) can be described with a 3D projective transformation (collineation). The number of parameters which describe the used entities are as follows.

- 3D point can be described with a 4-vector determined up to a scale. The degree of freedom is 3.
- 3D line can be described with a 6-vector determined up to a scale and a constraint (Plücker). The degree of freedom is 4.
- 3D projective transformation can be described with a 4x4 matrix determined up to a scale. The degree of freedom is 15.

Using these values the minimum number of entities to determine the invariant(s) is:

- 6 points yield $6 \times 3 - 15 + 0 = 3$ independent invariant
- 4 points and a line yield $(4 \times 3 + 4) - 15 + 0 = 1$ independent invariant
- 2 points and 3 lines yield $(2 \times 3 + 3 \times 4) - 15 + 0 = 3$ independent invariants
- 3 points and 2 lines yield $(3 \times 3 + 2 \times 4) - 15 + 0 = 2$ independent invariants
- 4 lines yield $4 \times 4 - 15 + 1 = 2$ independent invariants

The basic element of the projective invariants is the cross ratio and its generalizations for higher dimensions, see (A12), (A14) and (A15). In the following, using the different geometric configurations to calculate invariants, it is supposed that the elements are in general positions. Apart from the trivial degenerate cases, the nontrivial configurations will be determined.
An invariant could be undetermined, if one or more determinants are zero. This means coincident point(s) and/or lines. All of these cases are eliminated from further investigation.
Invariants of 6 points
As shown in (2) and also e.g. in (Quan, 1995), the number of independent solutions is 3. Using the ratio of product of determinants, a possible combination of independent invariants are:

$$I_1 = \frac{|\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_5| \cdot |\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_4\mathbf{Q}_6|}{|\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_6| \cdot |\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_4\mathbf{Q}_5|}, I_2 = \frac{|\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_5| \cdot |\mathbf{Q}_1\mathbf{Q}_3\mathbf{Q}_4\mathbf{Q}_6|}{|\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_6| \cdot |\mathbf{Q}_1\mathbf{Q}_3\mathbf{Q}_4\mathbf{Q}_5|}$$

$$I_3 = \frac{|\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_5| \cdot |\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_4\mathbf{Q}_6|}{|\mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_6| \cdot |\mathbf{Q}_2\mathbf{Q}_3\mathbf{Q}_4\mathbf{Q}_5|}$$

There are many ways to create a geometric configuration to represent the situation from which it is possible to calculate the cross ratio. Taking two points $\mathbf{Q}_1$ and $\mathbf{Q}_2$ as the axis, and using the remaining points $\mathbf{Q}_i$, $i=3,4,5,6$, four planes (pencil of planes) can be formed. The cross ratio of these planes can be determined as the cross ratio of points created as the intersection of these planes with an arbitrary line not intersecting the axis.

Invariant of 4 points and a line

Let $\mathbf{Q}_{L,i}$, $i=1,2$ denote two arbitrary distinct points on the line. In this case the invariant in the determinant form is:

$$I=\frac{|\mathbf{Q}_{L,1}\mathbf{Q}_{L,2}\mathbf{Q}_1\mathbf{Q}_3|\cdot|\mathbf{Q}_{L,1}\mathbf{Q}_{L,2}\mathbf{Q}_2\mathbf{Q}_4|}{|\mathbf{Q}_{L,1}\mathbf{Q}_{L,2}\mathbf{Q}_1\mathbf{Q}_4|\cdot|\mathbf{Q}_{L,1}\mathbf{Q}_{L,2}\mathbf{Q}_2\mathbf{Q}_3|}$$

The geometrical situation is similar to the 6 point case, but the axis of the pencil of planes is the line.

Invariants of 3 points and 2 lines

Let the two lines be denoted by $\mathbf{L}$ and $\mathbf{K}$, and $\mathbf{Q}_{L,i}, \mathbf{Q}_{K,i}$, $i=1,2$ are two points on these lines, respectively. As shown above, there must be two independent invariants for this configuration.

$$I_1=\frac{|\mathbf{Q}_{L,1}\mathbf{Q}_{L,2}\mathbf{Q}_1\mathbf{Q}_2|\cdot|\mathbf{Q}_{K,1}\mathbf{Q}_{K,2}\mathbf{Q}_1\mathbf{Q}_3|}{|\mathbf{Q}_{L,1}\mathbf{Q}_{L,2}\mathbf{Q}_1\mathbf{Q}_3|\cdot|\mathbf{Q}_{K,1}\mathbf{Q}_{K,2}\mathbf{Q}_1\mathbf{Q}_2|},\ I_2=\frac{|\mathbf{Q}_{L,1}\mathbf{Q}_{L,2}\mathbf{Q}_1\mathbf{Q}_2|\cdot|\mathbf{Q}_{K,1}\mathbf{Q}_{K,2}\mathbf{Q}_2\mathbf{Q}_3|}{|\mathbf{Q}_{L,1}\mathbf{Q}_{L,2}\mathbf{Q}_2\mathbf{Q}_3|\cdot|\mathbf{Q}_{K,1}\mathbf{Q}_{K,2}\mathbf{Q}_1\mathbf{Q}_2|}$$

A possible geometric configuration to determine the cross ratio is the three planes formed by $\mathbf{L}$ and points $\mathbf{Q}_i$, $i=1,2,3$, and the plane generated by the three points. Using the line $\mathbf{K}$ to cut through these planes, the intersection of the line and the planes gives four points. The other invariant can be determined by interchanging the role of the lines.

Invariants of 2 points and 3 lines

Let $\mathbf{L}_i$, $i=1,2,3$ and $\mathbf{Q}_j$, $j=1,2$, be the three lines and two points, respectively. Geometrically, four planes could be defined from a pair of a point and a line. For example, let the four planes: $(\mathbf{L}_1,\mathbf{Q}_1)$, $(\mathbf{L}_1,\mathbf{Q}_2)$, $(\mathbf{L}_2,\mathbf{Q}_1)$ and $(\mathbf{L}_2,\mathbf{Q}_2)$. The remaining line $\mathbf{L}_3$ intersects these planes and the four intersection points on the line determine the cross ratio. The other two invariants could be calculated using lines 1,3 and 2,3 in plane definition.

Invariants of 4 lines

Let $\mathbf{L}_i$, $i=1,2,3,4$ be the four lines. This configuration has $4\times4-15+1=2$ projective invariants, because there is an isotropy subgroup of any collineation of 3D projective space that leaves the four lines in place (Hartley, 1992). Algebraically the invariants can be written as:

$$I_1=\frac{|\mathbf{Q}_{1,1}\mathbf{Q}_{1,2}\mathbf{Q}_{2,1}\mathbf{Q}_{2,2}|\cdot|\mathbf{Q}_{3,1}\mathbf{Q}_{3,2}\mathbf{Q}_{4,1}\mathbf{Q}_{4,2}|}{|\mathbf{Q}_{1,1}\mathbf{Q}_{1,2}\mathbf{Q}_{3,1}\mathbf{Q}_{3,2}|\cdot|\mathbf{Q}_{2,1}\mathbf{Q}_{2,2}\mathbf{Q}_{4,1}\mathbf{Q}_{4,2}|}\ \mathrm{I}_2=\frac{|\mathbf{Q}_{1,1}\mathbf{Q}_{1,2}\mathbf{Q}_{2,1}\mathbf{Q}_{2,2}|\cdot|\mathbf{Q}_{3,1}\mathbf{Q}_{3,2}\mathbf{Q}_{4,1}\mathbf{Q}_{4,2}|}{|\mathbf{Q}_{1,1}\mathbf{Q}_{1,2}\mathbf{Q}_{3,1}\mathbf{Q}_{3,2}|\cdot|\mathbf{Q}_{2,1}\mathbf{Q}_{2,2}\mathbf{Q}_{4,1}\mathbf{Q}_{4,2}|}$$

where $\mathbf{Q}_{i,j}$ denotes the j'th point on the line $\mathbf{L}_i$.

3.2 Projective and permutation Invariants

It is shown in (A13), that there are six possible values for the cross ratio for four collinear points. Using higher dimensional configurations, the situation is worse, 6 points has 6!=720

possible labeling. Therefore in order to use the invariants for indexing in the object database, the complexity of the query must be reduced. This means that the effect of labeling (permutations of the geometric entities) must be eliminated.
As it was shown previously, the invariants of different geometric configurations of points and lines can be written as the ratio of product of determinants. According to the simplest generalization of the form, at least $N+3$ points required in an N -dimensional space, thus

$$I(\mathbf{Q}_1,\mathbf{Q}_2,\ldots,\mathbf{Q}_N,\mathbf{Q}_{N+1},\mathbf{Q}_{N+2},\mathbf{Q}_{N+3})=\frac{|\mathbf{Q}_1\mathbf{Q}_2\cdots\mathbf{Q}_N\mathbf{Q}_{N+2}|\cdot|\mathbf{Q}_1\mathbf{Q}_2\cdots\mathbf{Q}_{N+1}\mathbf{Q}_{N+3}|}{|\mathbf{Q}_1\mathbf{Q}_2\cdots\mathbf{Q}_N\mathbf{Q}_{N+3}|\cdot|\mathbf{Q}_1\mathbf{Q}_2\cdots\mathbf{Q}_{N+1}\mathbf{Q}_{N+2}|}.$$

It can be seen, that in this case the changing of the labeling of the first $N-1$ points leaves the value of the invariant intact (the sign changes of the four determinants cancel each other), the permutation of the last four points yields the six different values. Therefore the permutations inside the invariant can be separated as

$$I(\pi(\mathbf{Q}_1,\ldots,\mathbf{Q}_N,\mathbf{Q}_{N+1},\mathbf{Q}_{N+2},\mathbf{Q}_{N+3}))=I(\pi_1(\mathbf{Q}_1,\ldots,\mathbf{Q}_{N-1})\pi_2(\mathbf{Q}_N,\mathbf{Q}_{N+1},\mathbf{Q}_{N+2},\mathbf{Q}_{N+3}))$$

where π denotes the permutations of the elements. Interchanging the elements between π_1 and π_2 yields other invariants. Putting together, the projective and permutation invariants must fulfill two requirements:

- *Problem 1*: Eliminate the effect of the six possible values of the cross ratio. This can be accomplished using algebraic or stereographic permutation invariants.
- *Problem 2*: Eliminate the effect of interchanging the elements between π_1 and π_2.

Permutation invariants for cross ratio

In the solutions proposed by (Meer et al, 1998), (Csurka & Faugeras, 1999), the elimination of the effect of the different labeling inside the cross ratio is achieved in an algebraic way using higher order symmetric polynomials. The developed method follows a different method, applies a stereographic projection and a periodic function to give a solution for *Problem 1*.

Stereographic permutation invariants for cross ratio

As it can be seen in Fig. 1 (left), the plot of the six possible permutations of the cross ratio is symmetrical to the value 0.5 and (projectively) ∞. By pairs equating the three basic functions (occurs in cross-ratio) $\{x,1/x,1-x\}$ yields $x=1/x\rightarrow x=\pm1$ and $x=1-x\rightarrow x=0.5$, the mapping of these values could be calculated. (Note that the third possible combination $1/x=1-x$ does not give real solution.)

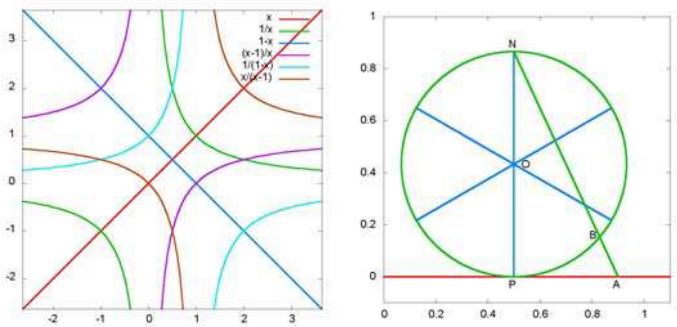

Fig. 1. Effect of permutations inside cross ratio (left), stereographic projection (right)

Considering Table 1 (note, that in projective manner the values ∞ and $-\infty$ represents the same) it can be concluded, that the key values of the six mappings are $(-1,0,0.5,1,2,\infty)$, because they form a closed set respecting to these mappings. In order to generate permutation invariants, application of such periodic function(s) is required that gives same value to the six possible combinations of the basic functions. This could be achieved in a two step process.

X	-1	0	0.5	1	2	∞
1/x	-1	∞	2	1	0.5	0
1-x	2	1	0.5	0	-1	∞

Table 1. Key values mappings inside cross ratio

Stereographic projection

In order to define a periodic function, the mapping of the infinite line (possible values of cross ratios) onto a circle is required. This could be achieved with the stereographic projection (used in the developed system, Fig. 1, right) or gnomonic projection. The parameters of the circle can be determined from the following constraints

- The values in the same pair must be mapped on the opposite side of the circle
- The infinity on the line must be mapped into the "north pole". Therefore the value 0.5 must be on the "south pole" (at point P).
- The arrangement of the (six) key values must be symmetrical.
- The mapping is continuous.

This yields, that the values $(0.5,1,2,\infty,-1,0)$ are mapped onto the angles $\sphericalangle(POB)=(0,\pi/3,2\pi/3,\pi,4\pi/3,5\pi/3)$, respectively. Note, that the $2\times\sphericalangle(PNB)=\sphericalangle(POB)$, because $\sphericalangle(POB)$ is the central angle and $\sphericalangle(PNB)$ is the respecting inscribed angle. The radius of the circle can be determined as $\tan(\sphericalangle(PNA))=\frac{A-P}{2r}\rightarrow r=\frac{A-P}{2\tan(\sphericalangle(PNA))}$. Substituting the values $(A=1,P=0.5,\ \sphericalangle(PNB)=\pi/6,\tan(\sphericalangle(PNB))=1/\sqrt{3})$ gives $r=\sqrt{3}/4$. The PDF (probability density function) of the stereographic permutation invariants is shown in Fig. 2.

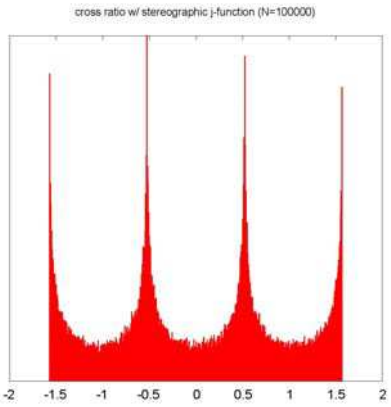

Fig. 2. Probability density function of stereographic permutation invariants

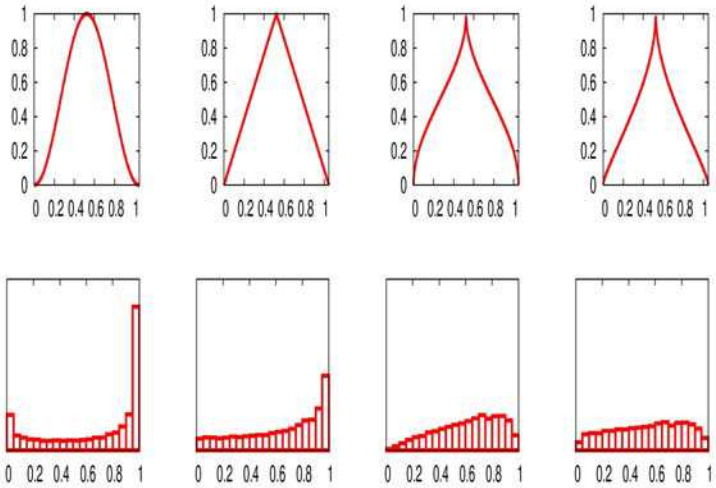

Fig. 3. The effect of nonlinear periodic functions (upper line) to the approximations of uniformly distributed PDF (lower line)

Application of a periodic function

Using the output of the stereographic mapping, the aim is to define a periodic function that fulfills the $J_p(I) = J_p(I + (k\pi/6)), k = 0,\ldots,5$ requirement. From the practical point of view, the outputs of the tested functions are mapped into $[0,1]$ interval. In order to apply a simple (Euclidean) distance function during the indexing, a nonlinear transformation must be defined such a way, that the output density must be close to the uniform one. Amongst the several possibilities, the following functions (whose period is $\pi/6$, against $\arcsin(\sin(x)) = x$ whose period is 2π) are tested (see Fig. 3 and note, the first row shows only one period of functions):

- $J_{p1} = \sin^2(3I)$
- $J_{p2} = (2/\pi)\,|\arcsin(\sin(3I))|$
- $J_{p3} = (2/\pi)\arcsin(\sqrt{(2/\pi)\,|\arcsin(\sin(3I))|})$
- $J_{p4} = 0.57(J_{pb}(I) - 6J_{pb}(I - (\pi/6))) + 0.86$, where

  $J_{pb} = (1/\pi)\arcsin(\sqrt{(1/\pi)\,|\arcsin(\sin(3I))|})$

Examining the PDF of the invariants applying the different functions, it can be seen that the J_{p4} gives the PDF closest (most similar) to the uniform distribution.

The output of the periodic function gives the solution to the *Problem 1*.

Elimination of the effect of element interchanges

The next step is to eliminate the effect of interchanging the elements between two permutation groups (giving solution to the *Problem 2*). The number of possible combinations is $\binom{N+3}{4}$. Therefore the permutation invariant is not a single value but a vector $\mathbf{J}$. In order

to remove the effect of the initial labeling of $N+3$ points, the vector must be sorted. The applicability of the following configurations is checked: 6 points, 1 line + 4 points, 2 lines + 3 points, 3 lines + 2 points, 4 lines, 5 lines.

Configuration: 6 points

In case of six points, interchanging the elements between the permutation groups yields the invariant vector

$$\mathbf{J} = \mathrm{S}\left(\begin{array}{c} \mathrm{J}(\mathrm{I}_1), \mathrm{J}(\mathrm{I}_2), \mathrm{J}\left(\frac{\mathrm{I}_1}{\mathrm{I}_2}\right), \mathrm{J}\left(\frac{\mathrm{I}_1-1}{\mathrm{I}_2-1}\right), \mathrm{J}\left(\frac{\mathrm{I}_2(\mathrm{I}_1-1)}{\mathrm{I}_1(\mathrm{I}_2-1)}\right), \mathrm{J}(\mathrm{I}_3), \mathrm{J}\left(\frac{\mathrm{I}_1}{\mathrm{I}_3}\right), \mathrm{J}\left(\frac{\mathrm{I}_1-1}{\mathrm{I}_3-1}\right), \mathrm{J}\left(\frac{\mathrm{I}_3(\mathrm{I}_1-1)}{\mathrm{I}_1(\mathrm{I}_3-1)}\right), \\ \mathrm{J}\left(\frac{\mathrm{I}_2}{\mathrm{I}_3}\right), \mathrm{J}\left(\frac{\mathrm{I}_2-1}{\mathrm{I}_3-1}\right), \mathrm{J}\left(\frac{\mathrm{I}_3(\mathrm{I}_2-1)}{\mathrm{I}_2(\mathrm{I}_3-1)}\right), \mathrm{J}\left(\frac{\mathrm{I}_3-\mathrm{I}_1}{\mathrm{I}_3-\mathrm{I}_2}\right), \mathrm{J}\left(\frac{\mathrm{I}_2(\mathrm{I}_3-\mathrm{I}_1)}{\mathrm{I}_1(\mathrm{I}_3-\mathrm{I}_2)}\right), \mathrm{J}\left(\frac{(\mathrm{I}_2-1)(\mathrm{I}_3-\mathrm{I}_1)}{(\mathrm{I}_1-1)(\mathrm{I}_3-\mathrm{I}_2)}\right) \end{array} \right) \tag{3}$$

where I_1, I_2 and I_3 are the invariants belonging to the permutation group $\pi_1(1,2)\pi_2(3,4,5,6)$ of points, $S(\cdot)$ denotes the sorting operator. The number of points in the configuration is six but the vector $\mathbf{J}$ has 15 elements. Therefore no one-to-one mapping exists between the points and the elements of the vector. Instead, the mapping exists between pairs of points and the respecting vector element. The first five elements in (3) depend on $(\mathbf{Q}_1\mathbf{Q}_i)$, $i=2,\ldots,6$, the next four depend on $(\mathbf{Q}_2\mathbf{Q}_i)$, $i=3,\ldots,6$, and so on. Finally the last element depends on $(\mathbf{Q}_5\mathbf{Q}_6)$. This means, that building a 6x6 table, according to the indexing with $\mathbf{J}$, the ordering of the points between two sets of respecting six point configurations can be determined in the following way.

We describe our concept for the 6-point case. Similar technique can be used for other feature combinations. The object database contains objects and the objects contain also points, from which different subsets containing 6 points can be built. The database contains Euclidean information belonging to the subset of points. From this information using the the homogeneous coordinates of the points the invariants can be computed. By using the nonlinear function J_{p4} the 15 (normalized) components of the vector $\mathbf{J}$ can be computed and sorted and the permutation $\mathbf{p}$ after sorting can be determined. This pair of $\mathbf{J}$ and $\mathbf{p}$ are precomputed and stored in the database before application. In the scene we can choose 6 point features and from their 3D projective coordinates we can determine another pair of $\mathbf{J}$ and $\mathbf{p}$ in a similar way during application. The basis for finding corresponding sets of points are the $\mathbf{J}$'s both in object database and scene. The $\mathbf{J}$'s are compared using Euclidean distance and a tolerance. Corresponding sets of points are marked and the collineation mapping points from scene into points from database is determined. This collineation makes it possible to map further points from the scene into database and check for correspondence. Thus the set of corresponding points belonging to the same object can be enlarged. In the success indices a and b identify the sets in database and scene, respectively. The main problem is that the order of the points in database and scene may be different. The details are as follows.

After sorting of the vectors $\mathbf{J}_a$ and $\mathbf{J}_b$, let $\mathbf{p}_a$ and $\mathbf{p}_b$ contain the permutation indices of the elements, therefore if $\mathbf{J}_a(i)=\mathbf{J}_b(i)$, $i=1,\ldots,15$, then element indexed by $\mathbf{p}_a(i)$ corresponds to $\mathbf{p}_b(i)$. Defining the vector $\mathbf{V}$ according to Table 2 yields that the pair $\mathbf{V}(\mathbf{p}_a(i))$ corresponds to $\mathbf{V}(\mathbf{p}_b(i))$, e.g. $\mathbf{V}(\mathbf{p}_a(1))=\mathbf{V}(6)=\{2,3\}$ corresponds to

$\mathbf{V}(\mathbf{p}_b(1)) = \mathbf{V}(13) = \{4,5\}$. Let $\mathbf{A}$ be a 6×6 (symmetric) table, where $\mathbf{A}(\mathbf{V}(\mathbf{p}_b(i))) = \mathbf{V}(\mathbf{p}_a(i))$. The *i*'th point in the set '*a*' corresponds to *j*'th point in the set '*b*', iff every element in the *i*'th row of $\mathbf{A}$ contains the index *j*.
For example a query from the scene into the database contains the sorted vector and permutation:

$$J_a = (0.0075 \quad 0.0247 \quad 0.0322 \quad 0.0344 \quad 0.0420 \quad 0.0667 \quad 0.3196 \quad 0.3269 \quad \cdots$$
$$0.4270 \quad 0.4341 \quad 0.7054 \quad 0.7185 \quad 0.7257 \quad 0.9517 \quad 0.9739)$$

$$p_a = (6 \quad 12 \quad 10 \quad 7 \quad 9 \quad 14 \quad 1 \quad 8 \quad 11 \quad 2 \quad 5 \quad 4 \quad 15 \quad 3 \quad 13)$$

The resulted entry from the database gives:

$$J_b = (0.0103 \quad 0.0338 \quad 0.0441 \quad 0.0468 \quad 0.0572 \quad 0.0909 \quad 0.3209 \quad 0.3309 \quad \cdots$$
$$0.4270 \quad 0.4367 \quad 0.7037 \quad 0.7219 \quad 0.7315 \quad 0.9513 \quad 0.9816)$$

$$p_b = (13 \quad 3 \quad 14 \quad 15 \quad 4 \quad 5 \quad 11 \quad 8 \quad 7 \quad 10 \quad 2 \quad 6 \quad 1 \quad 12 \quad 9)$$

The vector **V** is given in detailed form in Table 2.

I	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
V	1,2	1,3	1,4	1,5	1,6	2,3	2,4	2,5	2,6	3,4	3,5	3,6	4,5	4,6	5,6

Table 2. Possible pairings in the six points configuration

Using the permutation vectors $\mathbf{p}_a(1) = 6$ corresponds to $\mathbf{p}_b(1) = 13$, yields that pair 2,3 corresponds to pair 4,5. Write 2,3 into the position 4,5 (and 5,4) of the 6×6 table and continuing the process gives the results in Table 3.

	1	2	3	4	5	6
1	*	5,6	1,6	3,6	2,6	4,6
2	5,6	*	1,5	3,5	2,5	4,5
3	1,6	1,5	*	1,3	1,2	1,4
4	3,6	3,5	1,3	*	2,3	3,4
5	2,6	2,5	1,2	2,3	*	2,4
6	4,6	4,5	1,4	3,4	2,4	*

Table 3. Determine correspondences in six points configuration

Searching for the common elements row-wise (e.g. 6 in the first row in Table 3) gives the final pairings of the features: 1-6, 2-5, 3-1, 4-3, 5-2, 6-4.
A fault tolerant method is also developed. For example some numerically close elements in the corresponding vectors are swapped by sorting process, hence the table does not yield a valid solution, see the cells underlined in Table 4.
The solution to the problem is the following. Fill another 6x6 table from the original one such that the element in (*i,j*) contains the number of occurrences of *j*'th value in *i*'th row of the original table. Then repeat the following process:

1. Search for a maximum value in this new table. The row-column index gives the pairing.
2. Fill the row and the column with zeros of the pair already found. If the current maximum value is less than the desired parameter (tipically 4, tolerating only one mis-match), the pairing is not possible.

	1	2	3	4	5	6
1	*	5,6	1,5	3,5	4,5	2,5
2	5,6	*	1,6	3,6	4,6	2,6
3	1,5	1,6	*	**2,3**	1,4	1,2
4	3,5	3,6	2,3	*	3,4	1,3
5	4,5	4,6	1,4	3,4	*	2,4
6	2,5	2,6	1,2	**1,3**	2,4	*

1	2	3	4	5	6
1	1	1	1	5	1
1	1	1	1	1	5
4	2	1	1	1	1
1	1	5	1	1	1
1	1	1	5	1	1
2	4	1	1	1	1

Table 4. Determine correspondences in six points configuration (fault tolerant version)

Configuration: 1 line, 4 points

The calculation of the permutation invariant from the projective one is very simple, applying the function $\mathbf{J}(\cdot)$ to the only one projective invariant. But no method is currently known to determine pairings from permutation and projective invariants, therefore this type of configuration is *not used during indexing*.

Configuration: 2 lines, 3 points

As mentioned earlier, the geometric configuration for this case could be traced back to the five coplanar points case. Therefore the results of (Meer et al, 1998) could be used, namely interchanging the elements between the permutation groups yields the vector

$$\mathbf{J}_{2D} = S\left(J(I_1), J(I_2), J\left(\frac{I_1}{I_2}\right), J\left(\frac{I_1 - 1}{I_2 - 1}\right), J\left(\frac{I_2(I_1 - 1)}{I_1(I_2 - 1)}\right)\right) \tag{4}$$

The elements of the vector can be determined by exchanging the first element with the elements at $2,\ldots,5$, respectively.

But this is unnecessary, because the lines and points can be clearly distinguished, therefore the first element should only be exchanged with the second and the third one. Interchanging the two lines means applying $I \to 1/I$ mapping of the invariant (see the algebraic form). This means, that the permutation invariant vector should contain only

$$\mathbf{J} = S\left(J(I_1), J(I_2), J\left(\frac{I_1}{I_2}\right)\right).$$

If the pairing of the points and lines between two sets is required, the simplest solution is to calculate the vector defined in (4), because there is a one-to-one mapping between the five points and the five elements of $\mathbf{J}_{2D}$. A possible additional check is to pair points generated by line intersection with a similar one.

Configuration: 3 lines, 2 points

This configuration yields six planes, because a plane can be formed from a line and a point, where the point and the line are not coincident. In the projective 3D space the points and planes are dual to each other (principle of duality), therefore the results of the six points case can be used.

Configuration: 4 lines

The calculation of the permutation invariant from the projective one is simple, applying the appropriate function to the projective invariants. But no method is currently known to determine pairings from permutation and projective invariants, therefore this type of configuration is *not used during indexing*.

Configuration: 5 lines
In order to be able to use line only configuration, from which the pairing can be determined, compound configuration must be used. The simplest one is 5 lines in general position. From 5 lines five different 4-lines configuration can be extracted. A 4-lines configuration gives two independent invariants. Applying a function $f(J_{i,1}, J_{i,2}) \rightarrow J_i$, $i = 1,\ldots,5$ yields five different invariants. From these invariants the pairing could be determined.
Let the *i*'th configuration be the one from which the *i*'th line is excluded (1st configuration is built from lines 2,3,4,5, etc.). Let the unsorted 5-vectors be $\mathbf{J}_a$ and $\mathbf{J}_b$. Let the permutation vectors containing the output of the sorting be $\mathbf{p}_a, \mathbf{p}_b$, respectively. This means that the $\mathbf{J}_a(\mathbf{p}_a(i))$ invariant equals to $\mathbf{J}_b(\mathbf{p}_b(i))$, therefore the (eliminated) lines $\mathbf{p}_a(i)$, $\mathbf{p}_b(i)$ correspond to each other.

3.3 Object database
The aim of the application of the object database is to recognize known, predefined (previously stored) object(s) in the scene. The stored information in the database is the invariant vectors computed from *the 3D Euclidean description of the objects* represented by homogeneous coordinates as described in the previous section. During the query the input is computed from the output of the projective reconstruction of the scene. The two sets of invariants must be paired (matched) in order to determine the corresponding feature configurations. Some additional attributes also stored that is required during verification. The developed system uses different tables for each of the possible configurations (six points, etc.). The attributes are the name of the candidate object, type and id of the stored features and the permutation of the features. These values will be used in a later processing step (verification).
Metric definition and feature transformation
The usage of the database algorithms (indexing) requires the definition of a metric that describes the similarity of the feature combinations. A definition of a metric uses a distance function $d(\cdot)$ that describes the (dis)similarity of the elements between two sets, where $d = 0$ denotes identical configurations and the dissimilarity is larger as d increasing. Therefore d forms a metric, because i) d is a non-negative (real) number, ii) the relation is symmetrical, iii) fulfills the triangle inequality. In order to be able to compare the two feature sets, application of a feature transformation is required. This feature transformation maps the configuration properties into a D-dimensional vector space, where the distance between the vectors is defined. The distance between feature vectors must somehow correspond to the original (theoretical) distance between the features from them it was derived (eliminating false positives). Usually this means, that the distance between vectors is the lower bound of the original distance (this means that the small vector distance may yield dissimilar feature distance, but similar feature combinations always yield small vector distance). The properties used in the feature transformation are task dependent, in this case the feature configuration is described by an invariant vector defined in previous section. Therefore the feature transformation maps from features (described by its coordinates) into (vector)space of invariants. Many distance function can be created that fulfill the requirement of the definition. The most widely used functions can be described as

$$L(\cdot) = \left(\sum_{i=1}^{D} (a_i - b_i)^p \right)^{1/p}.$$

Using the different values of p yields the Manhattan metric $(p=1)$, Euclidean metric $(p=2)$ and maximum metric $(p=\infty)$. In the developed system the Euclidean metric is used.

Query into the database

The query process extracts those elements from the database that are closest to the querying element (exact matching is not probable due to noise during feature detection). This is the well-known nearest neighbors (kNN) problem. In our case the invariants are higher dimensional vector valued entities. The standard R-tree algorithm is very inefficient for higher dimensions (Moenne-Loccoz, 2005), due to the *curse of dimensionality*. The developed method uses *X*-tree (Berchtold et al., 1996). The query into the database extracts the closest candidates to the query vector (typically 2-5 are used). A tolerance is applied to eliminate the truly false matches. The remaining candidates are further processed in the verification step.

3.4 Verification

Because of the feature transformation the query eliminates only the false positives (those configurations, that are surely do not yield a valid answer to the query), the remaining candidates must be post-processed with a verification process. (Note: the query process should yield sufficiently small number of candidates in order to prevent the post-processing of the whole database.)

Collineation between 3D feature sets

Denote $\mathbf{H}$ the 4×4 matrix of the invertible linear transformation (collineation), $\mathbf{X}_i, \mathbf{Y}_i$ the 4-length coordinate vector of corresponding 3D homogeneous points. Let the corresponding line pair be $\mathbf{L}_i$ and $\mathbf{K}_i$, described by 4×4 $\mathbf{L}_i, \mathbf{K}_i$ skew-symmetric Plücker matrices, see (A5). Let $\mathbf{X}_{\mathbf{L}_i,r}$ be points on the line $\mathbf{L}_i$ and $\mathbf{Y}_{\mathbf{K}_i,s}$ be points on the line $\mathbf{K}_i$, respectively. Let $\mathbf{\Omega}_{\mathbf{K}_i,p}$ be planes that contains $\mathbf{K}_i$. If the $\mathbf{X}_i$ and $\mathbf{Y}_i$, $\mathbf{L}_i$ and $\mathbf{K}_i$ represent the same entities in different coordinate frames (related by $\mathbf{H}$), then the relation between them can be written into the form $\mathbf{Y}_i \sim \mathbf{H}\mathbf{X}_i$ and $\mathbf{K}_i \sim \mathbf{H}\mathbf{L}_i\mathbf{H}^T$, or using the entity-dependent scaling factors with equality $\mu_i\mathbf{Y}_i = \mathbf{H}\mathbf{X}_i$, $\nu_i\mathbf{K}_i = \mathbf{H}\mathbf{L}_i\mathbf{H}^T$. The aim is to determine $\mathbf{H}$ from a given set of point and line pairs in a noisy environment (LS solution is preferable), in a closed form. The solution must handle any number of combinations of points and lines. The unknowns are the 16 elements of the $\mathbf{H}$ matrix (and optionally the μ_i $(i=1,\ldots,n_P)$ scaling factors for points and the ν_i $(i=1,\ldots,n_L)$ scaling factors for lines).

Geometric solution

Using point and line pairs together, the equations contain the unknowns in quadratic or mixed form. Therefore the direct applications of these functions are not advisable. Instead geometric constraints are introduced in order to calculate the desired collineation. Let $\mathbf{H}$ be assumed in vector form

$$\mathbf{h}_{16\times1} = \begin{pmatrix} H(1,1) & \cdots & H(1,4) & \cdots & H(4,4) \end{pmatrix}^T = \begin{pmatrix} \mathbf{h}_1^T & \mathbf{h}_2^T & \mathbf{h}_3^T & \mathbf{h}_4^T \end{pmatrix}^T$$

Point-point relations

For points, the constraint equation is the scaling factor free algebraic distances

$$Y_i(a)\mathbf{X}_i\mathbf{h}_b^T - Y_i(b)\mathbf{X}_i\mathbf{h}_a^T = 0$$

where the pairs $\{a,b\}=\{4,1\},\{4,2\},\{4,3\},\{2,3\},\{3,1\},\{1,2\}$.
The part of the coefficient matrix belonging to this point pair is

$$\begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ -\Upsilon_i(4)\mathbf{X}_i & 0_4 & 0_4 & \Upsilon_i(1)\mathbf{X}_i \\ 0_4 & -\Upsilon_i(4)\mathbf{X}_i & 0_4 & \Upsilon_i(2)\mathbf{X}_i \\ 0_4 & 0_4 & -\Upsilon_i(4)\mathbf{X}_i & \Upsilon_i(3)\mathbf{X}_i \\ 0_4 & \Upsilon_i(2)\mathbf{X}_i & -\Upsilon_i(3)\mathbf{X}_i & 0_4 \\ -\Upsilon_i(1)\mathbf{X}_i & 0_4 & \Upsilon_i(3)\mathbf{X}_i & 0_4 \\ \Upsilon_i(1)\mathbf{X}_i & -\Upsilon_i(2)\mathbf{X}_i & 0_4 & 0_4 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{5}$$

Line-line relations
In order to eliminate the higher order members of the cost function, the line-type entities should be eliminated, points and planes relations must be used. The points on the line and planes, whose intersection is the given line can be extracted from the Plücker matrix of the line using SVD, see (A5) and (A25). Any linear combination of two points and two lines can be used as pairs instead of the original ones (resulted from SVD).
The two possible constraint types are:

- The transformed points $\mathbf{X}_{\mathbf{L}_i,r}$ should lie on the plane $\mathbf{\Omega}_{\mathbf{K}_i,s}$. Algebraically this means $\left(\mathbf{H}\mathbf{X}_{\mathbf{L}_i,r}\right)^T\mathbf{\Omega}_{\mathbf{K}_i,s}=0$ where $r,s=1,2$. The part of the coefficient matrix belongs to this configuration is

$$\begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ \mathbf{\Omega}_{\mathbf{K}_i,s}(1)\mathbf{X}_{\mathbf{L}_i,r} & \mathbf{\Omega}_{\mathbf{K}_i,s}(2)\mathbf{X}_{\mathbf{L}_i,r} & \mathbf{\Omega}_{\mathbf{K}_i,s}(3)\mathbf{X}_{\mathbf{L}_i,r} & \mathbf{\Omega}_{\mathbf{K}_i,s}(4)\mathbf{X}_{\mathbf{L}_i,r} \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{6}$$

A plane can be constructed from a transformed point $\mathbf{X}_{\mathbf{L}_i,r}$ and the line $\mathbf{\Omega}_{\mathbf{K}_i,s}$. If the point lies on the line, the plane equation must be invalid, $\mathbf{\Omega}=\mathbf{0}$. Using the representation in (A5), let $\mathbf{K}_i^T=\left(\mathbf{K}_{D,i}^T \quad \mathbf{K}_{O,i}^T\right)$, where $\mathbf{K}_{D,i}$ and $\mathbf{K}_{O,i}$ are 3-vectors. The plane can be generated using the matrix $\mathbf{\Lambda}_{\mathbf{K}_i}=\begin{pmatrix}\left[\mathbf{K}_{D,i}\right]_\times & \mathbf{K}_{O,i} \\ -\mathbf{K}_{O,i} & \mathbf{0}\end{pmatrix}$. Applying to the transformed point, the plane equation becomes $\mathbf{\Omega}_{\mathbf{K}_i,r}=\mathbf{\Lambda}_{\mathbf{K}_i}\left(\mathbf{H}\mathbf{X}_{\mathbf{L}_i,r}\right)$ where $r=1,2$. The part of the coefficient matrix belongs to this configuration is

$$\begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ \mathbf{\Lambda}_{\mathbf{K}_i}(1,1)\mathbf{X}_{\mathbf{L}_i,r} & \mathbf{\Lambda}_{\mathbf{K}_i}(1,2)\mathbf{X}_{\mathbf{L}_i,r} & \mathbf{\Lambda}_{\mathbf{K}_i}(1,3)\mathbf{X}_{\mathbf{L}_i,r} & \mathbf{\Lambda}_{\mathbf{K}_i}(1,4)\mathbf{X}_{\mathbf{L}_i,r} \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{7}$$

Estimation of $\mathbf{H}$
The equations (4), (5) and (6) yield linear constraints for the elements of the collineation $\mathbf{H}$. Collecting these coefficients into a matrix $\mathbf{A}$, the equations can be written into the form $\mathbf{A}\mathbf{h}=\mathbf{0}$. Applying an additional constraint $\|\mathbf{h}\|=1$ in order to avoid the trivial solution $\mathbf{h}=\mathbf{0}$, the problem can be solved in a closed form, using SVD, as the vector corresponding to the smallest singular value.

Thank You for previewing this eBook

You can read the full version of this eBook in different formats:

- HTML (Free /Available to everyone)
- PDF / TXT (Available to V.I.P. members. Free Standard members can access up to 5 PDF/TXT eBooks per month each month)
- Epub & Mobipocket (Exclusive to V.I.P. members)

To download this full book, simply select the format you desire below