Polytopes, Feasible Regions and Occlusions in the n-view Reconstruction Problem

David McKinnon[1], Barry Jones[2] and Brian C. Lovell[1]
[1]School of Information Techology and Electrical Engineering
Intelligent Real-Time Imaging and Sensing (IRIS) Group
University of Queensland
[2]Department of Mathematics
University of Queensland

Abstract

This paper asseses the question, given a arbitrary point in $\mathbb{P}^3$, can it be reconstructed by a given camera orbit? *We show that a solution to this problem can be found by intersecting the view frustums of the cameras in the sequence creating a polyhedron that bounds the area in $\mathbb{P}^3$ observed by all cameras. For a projective set of cameras this can be considered as an expansion of the cheiral inequalities. We also show an exception to this basic principle is encountered when the point in $\mathbb{P}^3$ is occluded. Thus giving a* weak condition for occlusion of an arbitrary point in $\mathbb{P}^3$.

1 Introduction

This work addresses the motivation to find the regions in $\mathbb{P}^3$ observed by two or more cameras in a given sequence. We will refer to this common region as the *feasible region* for the given set of cameras, it represents the part of space where depth fusion may occur through triangulation. This is an expansion of the basic theory of *cheirality* [3], where in this case the inequalities are expanded to constrain a point to lie with the view frustum of each camera of interest. This work is also partly inspired be the treatment of polyhedral models in the computer graphics literature [5]. The following subsections introduce the basic concepts and notation maintained throughout the rest of the paper.

1.1 Set Notation

The feasible region for n views $(\alpha_1, \ldots, \alpha_n)$ will be denoted as the set $\mathcal{S}^{\alpha_1 \cdots \alpha_n}$ of points (where $\mathcal{S}^{\alpha_1 \cdots \alpha_n} \in \mathbb{P}^3$) that project (un/occluded) into each of the n-images. The problem of finding the feasible region for a sequence of n-images has strong analogues with the set of the relevant subproblems of finding the intersection of 2 or more views $\alpha_a, \ldots, \alpha_b \in \{\alpha_1, \ldots, \alpha_n\}$. Where $\alpha_a, \ldots, \alpha_b$ depicts the choice of $2 \leq k \leq n$ unique views from the set of n, there are $\binom{n}{k}$ such choices. The notation for the $\binom{n}{k}$ subsets of $\mathcal{S}^{\alpha_1 \cdots \alpha_n}$ is given as $\mathcal{S}^{\alpha_1 \cdots \alpha_n}_{\alpha_a, \ldots, \alpha_b}$. Feasible regions and their $\binom{n}{k}$ *subregions* have the inclusion relation $\mathcal{S}^{\alpha_1 \cdots \alpha_n} \subset \mathcal{S}^{\alpha_1 \cdots \alpha_n}_{\alpha_a, \ldots, \alpha_b}$.

We can now answer our initial questions with the set notation. Firstly, which set of points in $\mathbb{P}^3$ can be reconstructed by the given camera orbit?

$$S^n_{\cup} \equiv \bigcup_{\substack{perm_k(\alpha_a, \ldots, \alpha_b) \\ 2 \leq k \leq n}} \mathcal{S}^{\alpha_1 \cdots \alpha_n}_{\alpha_a, \ldots, \alpha_b} \tag{1}$$

Secondly, which set of points in $\mathbb{P}^3$ can be reconstructed using *all* the given views from the camera orbit?

$$S^n_{\cap} \equiv \bigcap_{\substack{perm_k(\alpha_a, \ldots, \alpha_b) \\ 2 \leq k \leq n}} \mathcal{S}^{\alpha_1 \cdots \alpha_n}_{\alpha_a, \ldots, \alpha_b} \tag{2}$$

Section 2 will explore the inequalities that bound these sets and section 3 will present a rudimentary algorithm to compute the bounds for an arbitrary camera orbit in projective space. Section 4 discusses some applications of the *feasible regions*.

1.2 Projective Geometry of Linear Features in $\mathbb{P}^2$ and $\mathbb{P}^3$

This section will outline the notation and the basic building blocks for the representation of linear features in $\mathbb{P}^2$ and $\mathbb{P}^3$. The development of the ideas in this section is heavily influenced by the notation and stucture of the linear features presented in [6], also drawing some similarities to the oriented matching constraints in [7].

The first consideration when dealing with the multi-view geometry of linear features is their notation. Consistantly we will refer to features as any type of geometric object observed in a scene, be this points, lines or planes.

Table 1 summarises the notation and degrees of freedom (DOF) for the group of linear features in the projective plane ($[A, B, C] \in \mathbb{P}^2$).

Hyperplane	$\mathbb{P}^2$	$\mathbb{P}^{2*}$	DOF
Points	x^A	$x^{[A]} = \epsilon_{ABC}x^A = x_{[BC]}$	2
Lines	$x^{[AB]}$	$x^{[AB]} = \epsilon_{ABC}x^{AB} = x_C$	1

Table 1. Linear features and there duals in $\mathbb{P}^2$

Similarly, Table 2 summarises the notation and the DOF for linear features in projective space ($[a, b, c, d] \in \mathbb{P}^3$).

Hyperplane	$\mathbb{P}^3$	$\mathbb{P}^{3*}$	DOF
Points	x^a	$x^{[a]} = \epsilon_{abcd}x^a = x_{[bcd]}$	3
Lines	$x^{[ab]}$	$x^{[ab]} = \epsilon_{abcd}x^{ab} = x_{[cd]}$	2
Planes	$x^{[abc]}$	$x^{[abc]} = \epsilon_{abcd}x^{abc} = x_d$	1

Table 2. Linear features and there duals in $\mathbb{P}^3$

These tables demonstrate the process of dualization for linear feature types via the antisymmetrization operator $[\ldots]$. The antisymmetrization operator should be considered as a determinantal method to generate the algebra for linear features, by performing an alternating tensor contraction over the space/s to which the operator is applied [**?**, **?**]. The tensor notation will also convert directly into a computational scheme to evaluate the geometric entity in question.

An important aspect of the tensor notation, is the ease at which linear features can be manipulated to form geometrically intuitive combinations of each other. The tables given above demonstrate this concept clearly, where lines are constructed from two points and planes (in $\mathbb{P}^3$) are constructed from 3 points or a line and a point. This facet of the tensor notation (Grassmann-Cayley algebra) greatly increases it efficacy in solving for complex geometric configurations.

2 Feasible Regions

In this section we will introduce the basic camera model (pinhole camera) and describe the process of projecting it's boundaries to form the view frustum. We will then discuss the general intersection problem for 2 or more view frustum and give bounds for the number of differentally oriented solutions (cheirality), of which we will only be concerned with the positively oriented solutions.

We will round-off this section by discussing the *weak* condition for occlusion of an arbitrary point in $\mathbb{P}^3$. We call this condition weak because it not geometrically based, rather it is based on a lack of the observation of the point in the image, when the feasible region suggests that the point should be observable within the view frustums.

2.1 Camera Model and View Frustum

The pinhole camera model is the standard for modelling perspective distortion in image sequence. Perspective distortion is most prolific when a small focal length thus a greater Field of View (FOV), is employeed to model a scene that is both close to the camera and with a high depth variation.

Firslty, we must consider the projection operator (P^α_β) or *camera matrix* that denotes the projection of linear features from the scene to the image plane ($P^\alpha_\beta : \mathbb{P}^3 \to \mathbb{P}^3$). Table 3 summarises the range of projection operators for linear features.

Hyperplane	$\mathbb{P}^3$	$\mathbb{P}^{3*}$
Point	$x^A \sim P^A_a x^a$	-
Line	$x^{[AB]} \sim P^{[A}_{[a} P^{B]}_{b]} x^{[ab]}$	$x_{[BC]} \sim P^{[c}_{[B} P^{d]}_{C]} x_{[cd]}$
Plane	-	$x_A \sim P^d_A x_d$

Table 3. Projection operators for linear features

Generally it may be stated that $\lambda x^\alpha = P^\alpha_\beta x^\beta$, where λ is an arbitrary scale factor. The form of P^α_β for a regular point-to-point projection is,

$$P^\alpha_\beta \equiv [R^c_w | - R^c_w t^w] \tag{3}$$

where R^c_w is rotation from the center of the world reference frame to the orientation of the camera and t^w is the location of the camera in the world reference frame. This depiction of the camera (3), is defined in a Euclidean metric and also assumes calibration of the internal parameters. The more general projective uncalibrated model differs according to an affine ($K^A_{\hat{A}}$) and projective collineation ($H^{\hat{a}}_a$) [1]. Thus, the calibrated model for point projection is,

$$P^A_a \simeq K^A_{\hat{A}} P^{\hat{A}}_{\hat{a}} H^{\hat{a}}_a \tag{4}$$

where $K^A_{\hat{A}}$ represents the internal parameters of the camera in an affine collineation,

$$K^A_{\hat{A}} = \begin{pmatrix} \alpha_A & s & A_0 \\ 0 & \alpha_B & B_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

$\alpha_{A/B}$ encorporate the focal length f into scale factors for conversion between metric distances (mm) and pixels, s is the skew of the CCD imaging array and A_0 and B_0 are the coordinates of the cameras principal point [4].

From this point, no assumptions will be placed upon whether the cameras are un/calibrated. Instead, we will implicitly assume that problems dealing with calibrated cameras will present image coordinates as they would be read from the image plane ($A \in [0, \mathrm{cols}]$ and $B \in [0, \mathrm{rows}]$) and uncalibrated cameras present normalised image coordinates in a square centered fashion ($\hat{A} \in [-1, 1]$ and $\hat{B} \in [-1, 1]$) [2]. The ranges given for pixel coordinates typify the *bounds* for the camera. The projective (uncalibrated) assumption removes the ability to measure metric distances but can be made to preserve the orientation of an point-plane pair which we will see is the only requirement needed to build a proper view frustum.

The view frustum for the camera is found by an inverse projection of the bounds of the camera into the scene. That is, a projection of the lines that bound the image plane, into their corresponding planes in the scene. Figure 1 demonstrates the labelling of the vertices and edges for the standard camera.

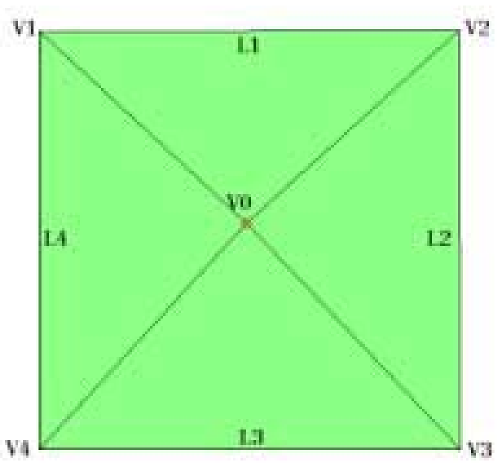

Figure 1. The labelling of the boundaries of projection for the standard view frustum.

Moving clockwise from the top left of the camera, the vertices V_1, V_2, V_3 and V_4 represent the corners of the image plane and V_0 is assigned to the cameras optical center (for finite cameras this is found as the nullvector of the camera matrix P^A_a or $\epsilon_{ABC} P^A_b P^B_c P^C_d \epsilon^{abcd}$). The polyhedron formed by the planes passing through V_0 and the joins of $V_1 \wedge V_2 = L_1, V_2 \wedge V_3 = L_2, V_3 \wedge V_4 = L_3$ and $V_4 \wedge V_1 = L_4$, gives a set of boundary conditions for a point in $\mathbb{P}^3$, where $L_\gamma = L^{[a_\alpha b_\beta]} = V^{[a_\alpha} V^{b_\beta]}$ are the lines (edges) joining the vertices of the image plane. A further set of inequalities should also impose the image plane (although this is a largely redundant constraint) as another boundary and the cheiral inequalities ($*$'s). These boundary conditions are expressed as a set of algebraic inequalities for a given point $x^d \in \mathbb{P}^3$, camera center V^{d_0} and plane-at-infinity $\pi^{[abc]}$.

$$\begin{array}{rrcl}
 & \lambda_1 L^{[a_1 b_2} V^{c_0} x^{d]} & > & 0 \\
 & \lambda_2 L^{[a_2 b_3} V^{c_0} x^{d]} & > & 0 \\
 & \lambda_3 L^{[a_3 b_4} V^{c_0} x^{d]} & > & 0 \\
 & \lambda_4 L^{[a_4 b_1} V^{c_0} x^{d]} & > & 0 \\
 & \lambda_5 V^{[a_1} V^{b_2} V^{c_3} x^{d]} & > & 0 \\
* & \lambda_6 \pi^{[abc} x^{d]} & > & 0 \\
* & \lambda_7 \pi^{[abc} V^{d_0]} & > & 0
\end{array} \qquad (6)$$

The scalars (λ_α) ensure the cheirality (orientation) of each plane with respect to the view frustum. This is achieved by making sure each point in the view frustum has a positive projective distance from the camera (only the sign is important). The last two inequalities ($*$'s) in (6) can be omitted from the boundary conditions if the scene points and cameras are know to be correctly oriented. Figure 2 shows some randomly generated positively oriented view frustums that have been clipped for display purposes. Note that not all of the cameras in this configuration are positioned to view the same part of space.

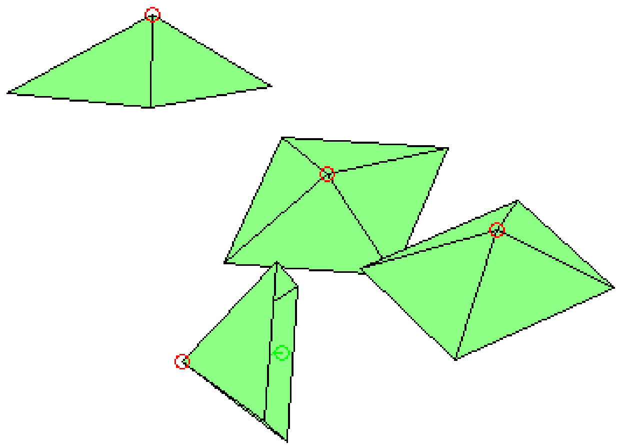

Figure 2. Four randomnly generated positively oriented view frustums for a square camera with a focal length of 1.

2.2 Feasible Regions and Cheiral Intersections

This section will move the discussion onto the case of multiple (n) cameras. Running out of meaningful indices, our first step will be to rewrite (6) in a symoblic vector notation, feeling comfortable that we have an effective means to compute the various vectors in the inequalities.

$$\begin{array}{rrcl} & \lambda_1 x^\top P_1^n & > & 0 \\ & \lambda_2 x^\top P_2^n & > & 0 \\ & \lambda_3 x^\top P_3^n & > & 0 \\ & \lambda_4 x^\top P_4^n & > & 0 \\ & \lambda_5 x^\top P_0^n & > & 0 \\ * & \lambda_6 x^\top \pi & > & 0 \\ * & \lambda_7 V_0^{n\top} \pi & > & 0 \end{array} \tag{7}$$

The superscript on the vectors denotes the image number and the subscript reflects the labelling (Figure 1) of the planes (P_β^α), points (V_β^α) and plane-at-infinity (π) where P_0^n is the n^{th} image plane. The inequalities given in (7) can now be considered to represent a valid set of points in the feasible region for the n-view reconstruction problem ($S_\cap^n$) and the entire set of points that are *reconstructable* ($S_\cup^n$) can be found as the union of the $\binom{n}{k}$ subproblems (where $2 \leq k \leq$ n).

This definition of the feasible regions for the n-view reconstruction problem is tractable, however we are compelled to find a more definitive statement for the regions that captures their geometric structure more precisely allowing visualisation, volume calculations and a more compact representation. This is where we introduce the language of polytopes.

Polytopes [8] allow us to represent the feasible regions as a convex hull of vertices (points), edges (lines) and facets (planes) in $\mathbb{P}^3$, thus providing a bounding polyhedra for the various linear programming problems implied by the application of the inequalities in (7). Since all planar intersections occur either before the plane-at-infinity or at the plane-at-infinity (for parallel planes) we will strip the last two inequalities ($*$'s) along with the 5^{th} (to cut down computation) from (7) leaving just the boundary conditions for the outer planes in the view frustums.

The bounding polyhedra will be constructed from the intersection of pairs of the four remaining planes from each camera (ie. lines $V_0^n \wedge V_\beta^n$ for camera n there is $\binom{4}{2}$ such lines for each camera) with the set of four remaining planes in each other camera (P_β^α where $\alpha \neq n$). In the general case there are two different ways that a line can intersect a plane. Firstly, if the points on the line and the plane are coplanar then their intersection will be the original line, or if the line and the plane are not coplanar then their intersection will result in a point. The points resulting from the line-plane intersections will form the vertices of the bounding convex hull representing the feasible region for a given set of cameras.

In summary we can consider the upper and lower bounds for the number of line-plane intersections for n-views (η_n) to be,

$$0 \leq \eta_n \leq 4 \times \binom{4}{2} \times \frac{n!}{(n-2)!} \tag{8}$$

the lower bound is the case of n-views with the same internal camera parameters all lieing in the same position with the same orientation, this is clearly a *critical configuration* where no depth can be recovered. The upper bound is formed by the general case given by n-views of a regular pinhole camera where the set of four lines from each camera must intersect each plane from each other camera. The intersection points will be positively and negatively oriented (ie. lie in front of the cameras and behind).

Of the collection of intersection points formed in the planes, only a certain number will be valid (feasible) points. Valid points are those that conform to the inequalities given in (7). This is to say that we are only interested in positively oriented intersection points that lie on the boundary of the n-view feasible region. An algorithm to find the intersection points that construct the bounding polyhedra will be considered in section 3.

2.3 A Weak Condition for Occlusions

The formulation for the feasible region given above will determine if a point lies within the intersection of the view frustums for the camera involved. However, this provides no indication as to whether the point of interest is occluded by some other object in the scene.

This is brings us to consider a statement about whether a point that lies in a valid feasible region for a set of cameras is *observable* or not. Firstly, a *strong* statement of observability requires a precise model of the scene in question and typical ray casting process (analogous to a image-to-world graphics engine) can resolve geometrically whether there exists a clear line of site from the point in the scene to the image.

Unfortunately, this strong statement of observability assumes that a complete 3D model has already been obtained. This is prohibitive for the purpose of building a 3D model from images, see we must seek another statement without this prerequisite.

The *weak* statement of observability requires that the assumed texture surrounding the point in the scene is observable in the expected location in image. This statement is clearly less precise, but practically quite an acceptable means to determined whether a point has been occluded. With the discussion of the feasible regions above and this statement of observability a robust method to locate the images where a feature is present ensues.

3 An Algorithm to Compute the n-view Intersection Polytope

We present a rudimentary algorithm to find the bounding polytope containing the set of points $S_\cap^n$ for n-views. The algorithm is geometrically intuitive, but far from optimal for task. For an optimal approach refer to [**?**].

The algorithm will follow along the lines of the discussion given in the previous section for isolating the line-plane intersection points. Then, application of the first 4 inequalities in (7) will reveal the superfulous intersection points which are not feasible. Of the remaining points which will now lie on the convex hull of the feasible region's polyhedra, edges must be selected to intersect points that lie only on a common plane of one of the view frustums. The proposed algorithm is given in Table (4).

	Algorithm for the polyhedra bounding $S_\cap^n$
1.	Find the η_n intersections, where $k = 2, \ldots, n$
1.1	Find the vertices $\Longrightarrow \epsilon_{abcd} x^{a_\gamma} = w_{[b_\alpha} y_{c_\beta} z_{d_\kappa]}$.
2.	Store the vertices that fulfill the first 4 inequalities (6).
3.	Find the edges from x^{a_γ} and the joins of x^{a_γ} in each plane.

Table 4. Proposed algorithm for calculating the bounding polyhedra for $S_\cap^n$

4 Applications

This section discusses some of the perceived applications of the concept of feasible regions in computer vision. It should be noted that this concept is very new and the authors are still considering the scope and application of the theory.

4.1 Object Querying

The motivation here is to be able to query a sequence of images of a static scene (and the corresponding 3D reconstruction of the scene) to see if a feature lies within the view frustum of a given camera or a given set of cameras.

Principally, we feel that this process may enhance the regular Structure From Motion (SFM) pipeline [**?**] with the ability to relocate lost or occluded tracks based on an expectancy for when they will return into the camera's field of view. Another related application would be to retrospectively update the tracking profile of a feature found in image α to all the previous images where the feature was observable.

This process could be used to great effect in the SFM pipeline for long sequences, where the camera completes a loop (ie. viewing the same area of the scene multiple times with some seperation in the exposure) which we will refer to as sub/sequence closure. In this case a very valuable alignment of the same set of features can be achieved to correct any drift in the egomotion estimation up until that point. This philosophy has been used with great success in closed turntable sequences.

4.2 Path Planning

Another perceived application of feasible regions is an autonomous robot fitted with either one or multiple cameras. If the robot is equipped with the mechanism to recognise certain objects in the scene (for our purpose we will imagine that these are polytope models), then the feasible region will allow the robots to plan a path around the object whereby the entire object of interest remains within the feasible regions of the camera at each time instant.

Complex polyhedral objects can be represented as the convex hull of vertices encompasing the poylhedra, if each of the vertices is visible un/occluded in each view of the camera/s views then the robot has maintained a suitable perspective on the object.

A robot fitted with cameras could then be instructed to model an entire area (at its own pace) making sure that it has gathered an adequate amount of information to fulfill the accuracy requirements of the mission.

4.3 Building Voxel Volumes

The final application we have considered is building an arbitrary indexed voxel volume given a sequence of camera motions. Currently, the authors are yet see an application of voxel volume modelling where the voxel space has not been predesignated to house the object of interest. We feel that if the voxel volume 3D reconstruction technique is to reach its full practical value, the voxel spaces must be defined purely by the path of the camera, not placed at a point in space as a priori to reconstruction.

Furhermore, information about the number of views that a feature is observable in and whether there are occlusions of the feature for a given path of the camera, is invaluable for the construction of the metric space and the subsequent optimisation problem for the building of the voxel volume.

5 Conclusions and Future Work

We have presented a novel idea that sets the boundary for points in space that are reconstructable by an arbitrary camera path. We have considered which points are reconstructable in the general case and which points are observable in every image. We have also shown that this formulation leads to a weak condition for occlusions.

As a matter of future work the authors would like to consider the quality of observation for a point and the subsequent quality of reconstruction for a point given multiple observations. The envisaged formulation is to consider the expected error in the observation of a feature in the image and the projection of this error into space forming a cone. The intersection of multiple error cones gives something akin to a set of geodesic probability surfaces, representing the likelihood of the corresponding 3D feature lieing in this part of space.

The introduction of such a concept would lead to a sophisticated method to determine the overall quality of reconstruction for the scene given a specific camera orbit. Then, points that are positioned poorly in the scene for reconstruction purposes can treated with a greater caution.

In this manor the quality of reconstruction can be set as an input parameter to the SFM pipeline and made to replicate the average depth fusion for human vision. This can be considered as the ideal quality of reconstruction for visualisation with VR glasses.

The authors are also looking into optimal algorithms to intersect the view frustums and create the resulting polyhedron.

6 Acknowledgements

The authors would like to thank Carlos Leung and Ben Appleton for many interesting discussions on the development of the ideas presented in this paper. Also we would like to thank Bob Andrews for his assistance in editing this paper.

References

[1] R. I. Hartley. Euclidean reconstruction from uncalibrated views. *Applications of Invariance in Computer Vision - LNCS*, 825:237–256, 1994.

[2] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997.

[3] R. I. Hartley. Chirality. *Int. Journal of Computer Vision*, 26(1):41–61, 1998.

[4] R. I. Hartley and A. Zisserman. Multiple view geometry in computer vision. Cambridge University Press, 2000.

[5] S. Savchenko. *3D Graphics Programming : Games and Beyond*. SAMS Publishing, 2000.

[6] W. Triggs. The geometry of projective reconstruction i: Matching constraints and the joint image. *Int. Conf. on Computer Vision*, pages 338–343, 1995.

[7] T. Werner and T. Pajdla. Oriented matching constraints.

[8] G. M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics, Springer-Verlag, 1994.