Single View Computer Vision in Polyhedral World: Geometric Inference and Performance Characterization

Mingzhou Song, Aiwen Guo and Robert M. Haralick
University of Washington
Department of Electrical Engineering
Intelligent Systems Laboratory
Seattle, WA 98195-2500, U.S.A.
{msong,guo,haralick}@isl.ee.washington.edu

Abstract

An algorithm for making consistent 2-D to 3-D geometric inference in polyhedral world using one perspective line drawing is described. Hypotheses are made on the internal angles of visible faces. The normals to the face planes are then determined. Valid normals lead to the reconstruction of the 3-D polyhedral world up to a scale factor. The performance of the algorithm is verified by using covariance matrix propagation. The experimental results show satisfactory performance.

The general propagation formulae for the covariance matrix of both input and output variables are also derived.

1. Introduction

Given only one 2-D perspective projection line drawing of a 3-D polyhedral world and the camera calibration, without any other hints except that the faces of polyhedra are planar, what can be inferred about the polyhedral world, in terms of the geometric properties of the polyhedra? How accurate is the inference? These are two aspects of a classical computer vision problem. The goal is to find a possible explanation of the polyhedral world which would be consistent with the observed 2-D image.

Liebowitz *et al* [3] and Shum *et al* [4] proposed methods to solve the single view computer vision problem. Both authors took advantage of geometric regularity in the real world, such as parallelism and orthogonality.

In our approach, the polyhedra can have general shapes, not limited to those with parallelism or orthogonality.

Hypotheses are made on the internal angles of visible faces. The normals to the face planes are then determined. Valid normals lead to the reconstruction of the 3-D polyhedral world up to a scale factor.

The validity of the hypotheses is tested using the covariance matrix associated with the solution, which is derived analytically starting from the covariance matrix of the observed quantities and is propagated through each inference step. The performance validation is important especially when non-deterministic algorithms are involved because the behavior of such algorithms can be at most probabilistically predicted, but not logically.

2. Propagation of Covariance Matrix of Both Input and Output Variables

$\Sigma_{\Delta\vec{X}} \in \mathbb{R}^{n\times n}$ denotes the covariance matrix of $\vec{X}$ and $\Sigma_{\Delta\vec{\Theta}} \in \mathbb{R}^{m\times m}$ for $\vec{\Theta}$. The covariance propagates from $\Sigma_{\Delta\vec{X}}$ to $\Sigma_{\Delta\vec{\Theta}}$. Haralick [1] summarized the methodology of covariance propagation from observed data vector $\vec{X} \in \mathbb{R}^n$ to inferred unobserved parameter vector $\vec{\Theta} \in \mathbb{R}^m$.

However, there are situations where we might be interested in the covariance matrix $\Sigma_{\vec{\Theta},\vec{X}} \in \mathbb{R}^{(n+m)\times(n+m)}$, which gives the covariance between $\vec{\Theta}$ and $\vec{X}$ as well as $\Sigma_{\Delta\vec{\Theta}}$ and $\Sigma_{\Delta\vec{X}}$. One such situation happens when inference is not being made in the first step but the middle steps and $\vec{X}$ is an inferred quantity from previous steps rather than a directly observed quantity. $\vec{\Theta}$ is further inferred from $\vec{X}$. Since they are all inferred quantities, it is likely that the covariance between $\vec{\Theta}$ and $\vec{X}$ would be of interest. In the following sections, we will derive the closed-form formula to calculate the covariance matrix $\Sigma_{\vec{\Theta},\vec{X}}$. The same assumptions as in [1] have to be made so that the first order approximation is good enough.

2.1. Explicit Function $\vec{\Theta} = \mathbf{F}(\vec{X})$ Is Known

Given the explicit function $\vec{\Theta} = \mathbf{F}(\vec{X}) : \mathbb{R}^n \to \mathbb{R}^m$, $\vec{\Theta}$ is calculated by evaluating $\mathbf{F}(\vec{X})$, which is a vector function in general. Taking the first order Taylor series expansion, we get $\Delta\vec{\Theta} \approx \frac{d\mathbf{F}(\vec{X})^T}{d\vec{X}}\Delta\vec{X}$. So the covariance matrix can be found as

$$\Sigma_{\Delta\vec{\Theta},\Delta\vec{X}} \approx \begin{bmatrix} \mathbf{G}^T\Sigma_{\Delta\vec{X}}\mathbf{G} & \mathbf{G}^T\Sigma_{\Delta\vec{X}} \\ \Sigma_{\Delta\vec{X}}\mathbf{G} & \Sigma_{\Delta\vec{X}} \end{bmatrix} \quad (1)$$

where $\mathbf{G} = \frac{d\mathbf{F}(\vec{X})}{d\vec{X}}$. The upper-left sub-matrix is $\Sigma_{\Delta\vec{\Theta}}$.

2.2. Unconstrained Optimization with Objective Function $F(\vec{X},\vec{\Theta})$

$F(\vec{X},\vec{\Theta}) : (\mathbb{R}^n,\mathbb{R}^m) \to \mathbb{R}^+ \cup \{0\}$ is a nonnegative scalar function of $\vec{X}$ and $\vec{\Theta}$. $\vec{\Theta}$ is the optimal solution to the minimization problem: $\min_{\vec{\Theta}\in\mathbb{R}^m} F(\vec{X},\vec{\Theta})$. From [1], we know $\Delta\vec{\Theta} \approx -\left(\frac{\partial\vec{g}}{\partial\vec{\Theta}}\right)^{-1}\frac{\partial\vec{g}^T}{\partial\vec{X}}\Delta\vec{X}$, where $\vec{g} = \frac{\partial F(\vec{X},\vec{\Theta})}{\partial\vec{\Theta}}$.

It follows by definition that $\Sigma_{\Delta\vec{\Theta},\Delta\vec{X}}$ is

$$\begin{bmatrix} H^{-1}M^T\Sigma_{\Delta\vec{X}}MH^{-1} & -H^{-1}M^T\Sigma_{\Delta\vec{X}} \\ -\Sigma_{\Delta\vec{X}}MH^{-1} & \Sigma_{\Delta\vec{X}} \end{bmatrix} \quad (2)$$

where $H = \frac{\partial\vec{g}}{\partial\vec{\Theta}}$ and $M = \frac{\partial\vec{g}}{\partial\vec{X}}$. The upper-left sub-matrix is $\Sigma_{\Delta\vec{\Theta}}$.

2.3. Constrained Optimization with Objective Function $F(\vec{X},\vec{\Theta})$ and constraints $\mathbf{s}(\vec{\Theta})$

$F(\vec{X},\vec{\Theta}) : (\mathbb{R}^n,\mathbb{R}^m) \to \mathbb{R}^+ \cup \{0\}$ is a nonnegative scalar function of $\vec{X}$ and $\vec{\Theta}$. $\mathbf{s}(\vec{\Theta}) = 0$ represents a set of equations constraining $\vec{\Theta}$. These constraints can be equalities or inequalities. $\vec{\Theta}$ is the optimal solution to the minimization problem $\min_{\mathbf{s}(\vec{\Theta})} F(\vec{X},\vec{\Theta})$. From [1], we know

$$\begin{bmatrix} \frac{\partial\vec{g}}{\partial\vec{\Theta}} & \frac{\partial\mathbf{s}}{\partial\vec{\Theta}} \\ \frac{\partial\mathbf{s}^T}{\partial\vec{\Theta}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\vec{\Theta} \\ \Delta\vec{\Lambda} \end{bmatrix} \approx \begin{bmatrix} -\frac{\partial\vec{g}^T}{\partial\vec{X}} \\ \mathbf{0} \end{bmatrix} \Delta\vec{X} \quad (3)$$

where $\vec{g} = \frac{\partial F(\vec{X},\vec{\Theta})}{\partial\vec{\Theta}}$ and $\vec{\Lambda}$ is the Lagrangian multiplier vector.

Transforming Eq 3 to

$$\begin{bmatrix} \frac{\partial\vec{g}}{\partial\vec{\Theta}} & \frac{\partial\mathbf{s}}{\partial\vec{\Theta}} & \mathbf{0} \\ \frac{\partial\mathbf{s}^T}{\partial\vec{\Theta}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \Delta\vec{\Theta} \\ \Delta\vec{\Lambda} \\ \Delta\vec{X} \end{bmatrix} \approx \begin{bmatrix} -\frac{\partial\vec{g}^T}{\partial\vec{X}} \\ \mathbf{0} \\ \mathbf{I} \end{bmatrix} \Delta\vec{X}$$

and letting

$$C = \begin{bmatrix} \frac{\partial\vec{g}}{\partial\vec{\Theta}} & \frac{\partial\mathbf{s}}{\partial\vec{\Theta}} & \mathbf{0} \\ \frac{\partial\mathbf{s}^T}{\partial\vec{\Theta}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \quad D = \begin{bmatrix} -\frac{\partial\vec{g}^T}{\partial\vec{X}} \\ \mathbf{0} \\ \mathbf{I} \end{bmatrix}$$

we obtain the covariance matrix (C is symmetric)

$$\Sigma_{\Delta\vec{\Theta},\Delta\vec{\Lambda},\Delta\vec{X}} \approx C^{-1}D\Sigma_{\Delta\vec{X}}D^TC^{-1} \quad (4)$$

3. Geometric Inference and Covariance Propagation in the Polyhedral World

Given one 2-D perspective line-drawing of a polyhedral world, infer the 3-D positions and orientations of each polyhedron whose perspective projection can result in a perspective projected line-drawing statistically consistent with the observed 2-D perspective line-drawing.

Validate the inference procedure by comparing the covariance matrix derived analytically and the one estimated experimentally.

We avoid making hypotheses related to geometric properties change over space such as: *the 3-D position of a vertex.* We make hypotheses about intrinsic properties of a polyhedron, such as: *the length of one edge of each polyhedron*, and/or *the angle between two edges*, since these geometric properties usually do not change over space.

A strong property of polyhedron is that it has planar faces, which is a constraint in the problem. We start the inference procedure with hypotheses of angles and the length of one edge of each polyhedron. We repeat the hypothesizing procedure until consistency is attained.

Polyhedra having from some view the same topology but incongruent geometric structure might produce the same perspective images, even if we cannot come up with the "correct" values of the unknowns, we might arrive at a solution which produces the same perspective image. This is perfectly good up to the consistency requirement, since a single 2-D perspective image is not enough to make 3-D inference in general.

3.1. The Inference and Covariance Propagation Procedure

We start the inference procedure with a number of visible faces, edges and vertices in the given perspective image. These entities have been labeled and grouped.

We use the following notation to represent faces, edges and vertices: *visible faces* Π_k with normal $\vec{p}_k$ passing 3-D point $\mathbf{W}_k$, $k = 1,...,K$; *visible edges* L_i with direction cosine vector $\vec{b}_i$, two terminal points $\mathbf{P}_{i1}$ and $\mathbf{P}_{i2}$ and the observation plane normal $\vec{n}_i$, $i = 1,...,V$; *visible vertices* $\mathbf{P}_q$ with perspective projection $\mathbf{P}^*_q = (u_q, v_q, f)^T$, $q = 1,...,Q$.

To simplify the notation, we use:

$$\begin{array}{ll} \underline{\mathbf{P}} = (\mathbf{P}_1^T,\cdots,\mathbf{P}_Q^T)^T & \underline{\mathbf{b}} = (\vec{b}_1^T,\cdots,\vec{b}_V^T)^T \\ \underline{\mathbf{p}} = (\vec{p}_1^T,\cdots,\vec{p}_K^T)^T & \underline{\mathbf{W}} = (\mathbf{W}_1^T,\cdots,\mathbf{W}_K^T)^T \\ \underline{\mathbf{n}} = (\vec{n}_1^T,\cdots,\vec{n}_V^T)^T & \underline{\mathbf{uv}} = (u_1,v_1,\cdots,u_Q,u_Q)^T \end{array}$$

So, the start covariance matrix of the observed quantities is $\Sigma_{\underline{\mathbf{uv}}}$ and the one after propagation by inference is $\Sigma_{\underline{\mathbf{P}},\underline{\mathbf{b}},\underline{\mathbf{p}},\underline{\mathbf{W}}}$.

Step 1. Calculating the normal to the observation plane

Goal: The normal $\vec{n}_i$ to the observation plane of each edge L_i.
Observations: The 2-D perspective image $\mathbf{P}^*_{i1}\mathbf{P}^*_{i2}$ of each 3-D edge $\mathbf{P}_{i1}\mathbf{P}_{i2}, i = 1, ..., V$.
Inference: $\vec{n}_i = \frac{\mathbf{P}^*_{i1}\times\mathbf{P}^*_{i2}}{\|\mathbf{P}^*_{i1}\times\mathbf{P}^*_{i2}\|},\ i = 1, .., V$
Covariance Propagation: In this step, we have explicit formula for $\underline{\mathbf{n}}$, so by Eq 1 we know

$$\Sigma_{\underline{\mathbf{n}},\underline{\mathbf{uv}}} \approx \begin{bmatrix} \mathbf{G}_1^T\Sigma_{\underline{\mathbf{uv}}}\mathbf{G}_1 & \mathbf{G}_1^T\Sigma_{\underline{\mathbf{uv}}} \\ \Sigma_{\underline{\mathbf{uv}}}\mathbf{G}_1 & \Sigma_{\underline{\mathbf{uv}}} \end{bmatrix}$$

where

$$\mathbf{G}_1 = \frac{\partial\left[\left(\frac{\mathbf{P}^*_{11}\times\mathbf{P}^*_{12}}{\|\mathbf{P}^*_{11}\times\mathbf{P}^*_{12}\|}\right)^T, \cdots, \left(\frac{\mathbf{P}^*_{V1}\times\mathbf{P}^*_{V2}}{\|\mathbf{P}^*_{V1}\times\mathbf{P}^*_{V2}\|}\right)^T\right]^T}{\partial\underline{\mathbf{uv}}}$$

Step 2. From coplanar edges with known in-between angles on a face to infer the normal to the face

Suppose there are N coplanar edges $L_1, L_2, ..., L_N$, with unknown direction cosine $\vec{b}_1, \vec{b}_2, ..., \vec{b}_N$ on the face Π_k with unknown normal $\vec{p}_k$ at a position unknown.
Goal: Find the normal $\vec{p}_k$ to face Π_k, $k = 1, ..., K$.
Inferred Quantities from Previous Steps: $\vec{n}_1, \vec{n}_2, ..., \vec{n}_N$, with covariance matrix $\Sigma_{\vec{n}_1,...,\vec{n}_N}$.
Assumptions: Hypothesized M_k cosines of the angles between M_k $(M_k \leq N)$ pairs of the N coplanar lines. We write these assumptions as

$$|\gamma_1| = |\vec{b}_1\cdot\vec{b}_2|, \cdots, |\gamma_{M_k-1}| = |\vec{b}_{M_k-1}\cdot\vec{b}_{M_k}|, |\gamma_{M_k}| = |\vec{b}_{M_k}\cdot\vec{b}_1|$$

The angle consines that are most likely to be observed in the real world should be hypothesized [4]. *Inference:* For the i-th line L_i, we know: $\vec{b}_i \perp \vec{n}_i$ and $\vec{b}_i \perp \vec{p}_k$. So $\vec{b}_i = \frac{\vec{n}_i\times\vec{p}_k}{\|\vec{n}_i\times\vec{p}_k\|}$. Hence, for $i = 1, ..., M_k$,

$$|\gamma_i| = |\vec{b}_i\cdot\vec{b}_{i+1}| = \left|\frac{\vec{n}_i\times\vec{p}_k}{\|\vec{n}_i\times\vec{p}_k\|}\cdot\frac{\vec{n}_{i+1}\times\vec{p}_k}{\|\vec{n}_{i+1}\times\vec{p}_k\|}\right| \quad (5)$$

To solve for $\vec{p}_k$, we need in general two more equations, since we already have a constraint $\|\vec{p}_k\| = 1$. When there are more than two equations, i.e., $M > 2$, the equations can be solved by constrained optimization. Let

$$f(\vec{n}_i, \vec{n}_{i+1}, \vec{p}_k, \gamma_i) = \begin{cases} [(\vec{n}_i\times\vec{p}_k)\cdot(\vec{n}_{i+1}\times\vec{p}_k)]^2 \\ -(\gamma_i\|\vec{n}_i\times\vec{p}_k\|\|\vec{n}_{i+1}\times\vec{p}_k\|)^2, & \gamma_i \neq 0 \\ (\vec{n}_i\times\vec{p}_k)\cdot(\vec{n}_{i+1}\times\vec{p}_k), & \gamma_i = 0 \end{cases}$$

and the constrained optimization can be formulated as a non-linear least squares problem:

$$\min_{\|\vec{p}_k\|=1} F(\vec{n}_1, ..., \vec{n}_{M_k}, \vec{p}_k) = \min_{\|\vec{p}_k\|=1}\sum_{i=1}^{M_k} f^2(\vec{n}_i, \vec{n}_{i+1}, \vec{p}_k, \gamma_i)$$

The above problem can be solved numerically.

We make I random guesses for the initial value of each $\vec{p}_k$ and save the first J $(J \leq I)$ most optimal solutions for later use in hypothesis test. I and J are determined by training experiments.

After the numerical optimization, we check the consistency of the direction cosines of edges which can be derived from different paths. We select the best combination of the solutions of the normals which makes the direction cosines of edges most consistent. The consistency is checked by the summation of the squares of difference of direction cosines of edges which can be calculated by two different faces, that is, to find the best combination of $\vec{p}_k$, from the optimal and sub-optimal solutions which minimizes

$$\begin{aligned} \epsilon &= \sum_{i=1}^{V}\min\{\|\vec{b}_{i1}-\vec{b}_{i2}\|^2, \|\vec{b}_{i1}+\vec{b}_{i2}\|^2\} \\ &= \sum_{i=1}^{V}\min\left\{\left\|\frac{\vec{n}_i\times\vec{p}_{i1}}{\|\vec{n}_i\times\vec{p}_{i1}\|}-\frac{\vec{n}_i\times\vec{p}_{i2}}{\|\vec{n}_i\times\vec{p}_{i2}\|}\right\|^2,\right. \\ &\quad \left.\left\|\frac{\vec{n}_i\times\vec{p}_{i1}}{\|\vec{n}_i\times\vec{p}_{i1}\|}+\frac{\vec{n}_i\times\vec{p}_{i2}}{\|\vec{n}_i\times\vec{p}_{i2}\|}\right\|^2\right\} \end{aligned}$$

where $\vec{p}_{i1}$ and $\vec{p}_{i2}$ are the normals to the two visible faces which intersect at edge L_i.
Covariance Propagation: $\Sigma_{\underline{\mathbf{p}};\lambda_1,...,\lambda_K;\underline{\mathbf{n}}} \approx C^{-1}D\Sigma_{\underline{\mathbf{n}}}D^TC^{-1}$, where

$$\begin{aligned} C &= \begin{bmatrix} \left(\frac{\partial\vec{g}}{\partial\underline{\mathbf{p}}}\right)_{3K\times3K} & \left(\frac{\partial\vec{s}}{\partial\underline{\mathbf{p}}}\right)_{3K\times K} & \mathbf{0}_{3K\times3V} \\ \left(\frac{\partial\vec{s}}{\partial\underline{\mathbf{p}}}\right)^T_{K\times3K} & \mathbf{0}_{K\times K} & \mathbf{0}_{K\times3V} \\ \mathbf{0}_{3V\times3K} & \mathbf{0}_{3V\times K} & \mathbf{I}_{3V\times3V} \end{bmatrix} \\ D &= \begin{bmatrix} -\frac{\partial\vec{g}^T}{\partial\underline{\mathbf{n}}}_{3K\times3V} \\ \mathbf{0}_{K\times3V} \\ \mathbf{I}_{3V\times3V} \end{bmatrix} \\ \vec{g} &= \frac{\partial\sum_{k=1}^{K}F(\vec{n}_{k1}, ..., \vec{n}_{kM_k}, \vec{p}_k)}{\partial\underline{\mathbf{p}}}, \\ \vec{s} &= \left[\|\vec{p}_1\|^2-1, \cdots, \|\vec{p}_K\|^2-1\right]^T \end{aligned}$$

Step 3. From an edge on a face with known normal to infer the direction cosine of the edge

Goal: $\vec{b}_i,\ i = 1, ...V$
Inferred Quantities from Previous Steps: $\vec{p}_k,\ k = 1, ..., K$, where K is the number of visible faces. $\vec{n}_i,\ i = 1, ...V$
Inference: $\vec{b}_i = \frac{\vec{n}_i\times\vec{p}_k}{\|\vec{n}_i\times\vec{p}_k\|} \quad (i = 1)$

Step 4. From the length of an edge and its direction cosine to infer its 3-D position

Goal: The terminal points of the edge, $\mathbf{P}_{j1}, \mathbf{P}_{j2}$.

Observations: $\mathbf{P}_{j1}^* = (u_{j1}, v_{j1}, f)^T, \mathbf{P}_{j2}^* = (u_{j2}, v_{j2}, f)^T$
Inferred Quantities from Previous Steps: $\vec{b}_j, \quad j = 1$.
Assumptions: The length d of the j-th edge L_j of a polyhedron. The value of d does not affect the other inference results up to the scale.
Inference: See [2] p63 for details.

Step 5. From a known edge on a face to infer the 3-D position of the face

Goal: A point $\mathbf{W}_k$ on the face.
Inferred Quantities from Previous Steps: The k-th face normal $\vec{p}_k$. The terminal points of an edge $\mathbf{P}_{j1}, \mathbf{P}_{j2}$.
Inference: $\mathbf{W}_k = (\mathbf{P}_{j1} + \mathbf{P}_{j2})/2$, the geometric mid-point between $\mathbf{P}_{j1}$ and $\mathbf{P}_{j2}$, gives the best solution in the least square sense.

Step 6. From a known face to infer the 3-D positions of edges on this face

Goal: The terminal points of the i-th edge L_i, $\mathbf{P}_{i1}$ and $\mathbf{P}_{i2}$
Observations: $\mathbf{P}_{i1}^*$ and $\mathbf{P}_{i2}^*$
Inferred Quantities from Previous Steps: $\vec{p}_k$ and $\mathbf{W}_k$ of the face on which the edge is located.
Inference: $\mathbf{P}_{i1}$ and $\mathbf{P}_{i2}$ satisfy $(\mathbf{P} - \mathbf{W}_k)^T \vec{p}_k = 0$ and by perspective projection $\mathbf{P}_{i1} = \eta_{i1} \mathbf{P}_{i1}^*$ and $\mathbf{P}_{i2} = \eta_{i2} \mathbf{P}_{i2}^*$. Solving these equations, we obtain $\mathbf{P}_{i1} = \frac{\mathbf{W}_k^T \vec{p}_k}{\mathbf{P}_{i1}^{*T} \vec{p}_k} \mathbf{P}_{i1}^*$ and $\mathbf{P}_{i2} = \frac{\mathbf{W}_k^T \vec{p}_k}{\mathbf{P}_{i2}^{*T} \vec{p}_k} \mathbf{P}_{i2}^*$.

After the edge is known, go back to Step 5 to infer the position of the other face of the edge. Step 5 and step 6 form a recursive call loop. The recurrence keeps going until all the edges and faces of each polyhedron are resolved.

4. Experiment Results

We built a polyhedral world composed of a cuboid and a pyramid. Gaussian noise is applied to generate noisy observation. The noise is not required to be Gaussian as long as it has zero mean and finite variance. In Fig 1 shown, we printed out the perspective projection of the inferred 3-D world with noise standard deviation at 0.005 and 0.1.

All the non-deterministic behavior introduced by the algorithm comes from Step 2 – the numerical optimization whose result depends on initial guess. Hence, we concentrate our efforts on finding the covariance matrix about $\underline{\mathbf{p}}$ by Step 1 and 2. (The covariances of other geometric properties inferred from Step 3 to 6 are omitted since these steps are deterministic.) Corresponding entries in the analytical derived covariance matrix and the experiment-estimated one for $\underline{\mathbf{p}}$ show similar values and also form similar value patterns. (See *http://isl.ee.washington.edu/~msong/covariance.pdf*)

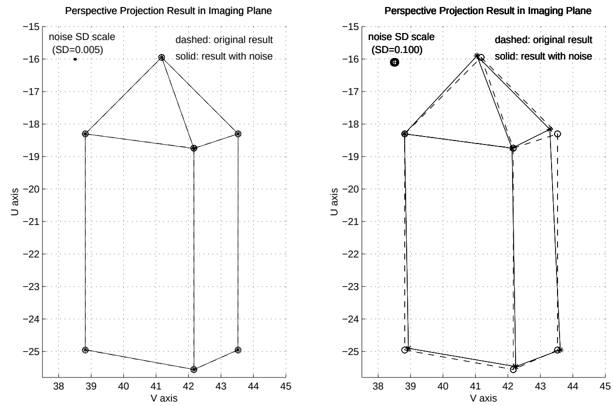

Figure 1. Perspective Projection of Inferred $3-D$ Polyhedral World

5. Conclusions

We gave a general approach to making geometric inference from 2-D to 3-D in a polyhedral world. We verified our implementation by comparing the estimated covariance matrix and the analytically computed covariance matrix. They showed consistent characteristics when the noise is small, additive and finite in variance.

The procedure outlined in this paper provides a way to begin for the covariance matrices of the observed 2-D perspective projection points and propagate these covariances to corresponding 3-D polyhedral vertices. The inferred 3-D positions are not those that have the smallest covariance. In another paper, we will discuss this more difficult problem.

For non-deterministic algorithms, *e.g.* some optimization problems whose solution depends on the choice of the initial value, the logical proof of correctness of a program is impossible. Off-line Monte Carlo testing has been the only way to test the correctness of the program. However, the on-line hypothesis testing using covariance matrix provides another way to safeguard the solution.

References

[1] R. M. Haralick. Propagating covariance in computer vision. *Intl. J. of PR & AI*, 10(5):561–72, Aug. 1996.
[2] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*, volume II. Addison-Wesley, Reading, MA, 1993.
[3] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. In *Proceedings EUROGRAPHICS*, volume 18, pages 39–50, September 1999.
[4] H.-Y. Shum, M. Han, and R. Szeliski. Interactive construction of 3d models from panoramic mosaics. In *CVPR'98*, pages 427–33, Santa Barbara, June 1998. IEEE.