A Survey of Geometric Vision

Kun Huang Yi Ma

Electrical & Computer Engineering Department

University of Illinois at Urbana-Champaign

Abstract

In this chapter we give a brief survey on basic geometric principles associated with machine vision. In particular, we introduce the basic theory and algorithms for the reconstruction of camera pose and scene structure from two views or more than two views. We discuss further reconstruction techniques from a single view, which utilize scene symmetry and nevertheless rely on the theory of multiple views. Since our survey focuses on only the very basics, references to more advanced studies will be provided after each topic. A few comprehensive examples of application of the algorithms are given at the end of the chapter.

1 Introduction

Vision is one of the most powerful sensing modalities. In robotics, machine vision techniques have been extensively used in applications such as manufacturing, visual servoing [30, 7], navigation [26, 50, 51, 9], and robotic mapping [58]. Here a main problem is how to reconstruct both the pose

of the camera and the 3-D structure of the scene. The reconstruction inevitably requires a good understanding about the geometry of image formation and 3-D reconstruction. In this chapter, we provide a survey for the basic theory and some recent advances in the geometric aspect of the reconstruction problem. Specifically, we introduce the theory and algorithms for reconstruction from two views [33, 29, 40, 67, 31], multiple views [40, 37, 38, 23, 10, 12], and a single view [25, 19, 1, 73, 74, 70, 3, 28]. Since this chapter can only provide a brief introduction to these topics, the reader is referred to the book [40] for a more comprehensive coverage.

Without any knowledge of the environment, reconstruction of a scene requires multiple images. This is because a single image is merely a 2-D projection of the 3-D world, for which the depth information is lost. When multiple images are available from different known viewpoints, the 3-D location of every point in the scene can be determined uniquely by triangulation (or stereopsis). However, in many applications (especially those for robot vision), the viewpoints are unknown either. Therefore, we need to recover both the scene structure and the camera poses. In computer vision literature, this is referred to as the "structure from motion" (SFM) problem. To solve this problem, the theory of *multiple-view geometry* has been developed (e.g., see [40, 37, 38, 67, 33, 23, 12, 10]). In this chapter, we introduce the basic theory of multiple-view geometry and show how it can be used to develop algorithms for reconstruction purposes. Specifically, for the two-view case, we introduce in Section 2 the *epipolar constraint* and the *eight-point structure from motion algorithm* [33, 29, 40]. For the multiple-view case, we introduce in Section 3 the *rank conditions* on *multiple-view matrix* [37, 38, 40, 27] and a multiple-view factorization algorithm [37, 40].

Since many robotic applications are performed in a man-made environment such as inside a building and around an urban area, much of prior knowledge can be exploited for a more efficient and accurate reconstruction. One kind of prior knowledge that can be utilized is the existence

of "regularity" in the man-made environment. For example, there exist many parallel lines, orthogonal corners, and regular shapes such as rectangles. In fact, much of the regularity can be captured by the notion of *symmetry*. It can be shown that with sufficient symmetry, reconstruction from a single image is feasible and accurate, and many algorithms have been developed (e.g., see [3, 70, 25, 1, 73, 40, 28, 1]). Interestingly, these symmetry-based algorithms in fact rely on the theory of multiple-view geometry [25, 70, 3]. Therefore, after the multiple-view case is studied, we introduce in Section 4 basic geometry and reconstruction algorithms associated with imaging and symmetry.

In the remainder of this section, we introduce in Section 1.1 basic notation and concepts associated with image formation that help the development of the theory and algorithms. It is not our intention to give in this chapter all the details about how the algorithms surveyed can be implemented in real vision systems. While we will discuss briefly in Section 1.2 a pipeline for such a system, we refer the reader to [40] for all the details.

1.1 Camera model and image formation

The camera model we adopt in this chapter is the commonly used pinhole camera model. As shown in Figure 1A, the camera comprises of a camera center o and an image plane. The distance from o to the image plane is the *focal length* f. For any 3-D point p in the opposite side of the image plane with respect to o, its image $\boldsymbol{x}$ is obtained by intersecting the line connecting o and p with the image plane. In practice, it is more convenient to use a mathematically equivalent model by moving the image plane to the "front" side of the camera center as shown in Figure 1B.

There are usually three coordinate frames in our calculation. The first one is the *world frame*,

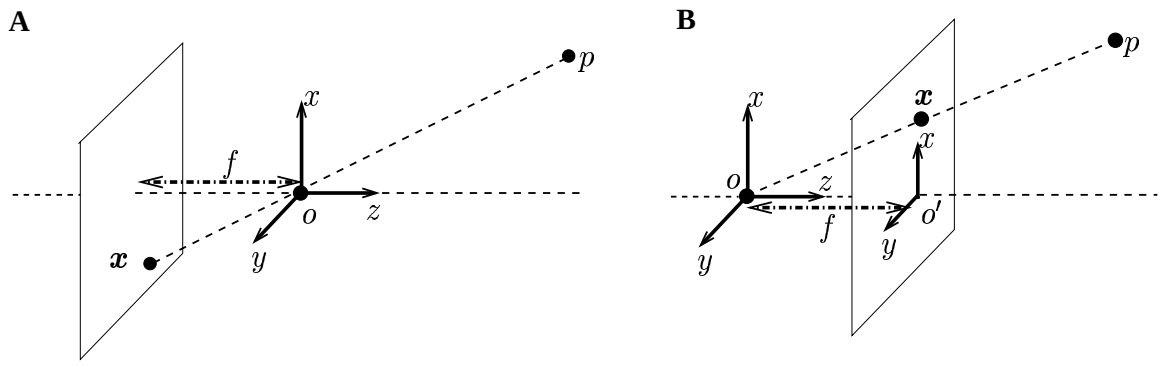

Figure 1: A: Pinhole imaging model. The image of a point p is the intersecting point $\boldsymbol{x}$ between the image plane and the ray passing through camera center o. The distance between o and the image plane is f. B: Frontal pinhole imaging model. The image plane is in front of the camera center o. An image coordinate frame is attached to the image plane.

also called *reference frame*. The description of any other coordinate frame is the motion between that frame and the reference frame. The second is the *camera frame*. The origin of the camera frame is the camera center and the z-axis is along the perpendicular line from o to the image plane as shown in Figure 1B. The last one is the two-dimensional *image frame*. For convenience we choose its origin o' as the projection of o on the image plane and set its x-axis and y-axis to be parallel to the x-axis and y-axis of the camera frame. As a convention, the focal length f is set to be of unit 1.

For points in 3-D space and 2-D image, we use *homogeneous coordinates*; i.e. a 3-D point p with coordinate $[x, y, z]^T \in \mathbb{R}^3$ is denoted as $\boldsymbol{X} = [x, y, z, 1]^T \in \mathbb{R}^4$, and an image point with coordinate $[x, y]^T \in \mathbb{R}^2$ is denoted as $\boldsymbol{x} = [x, y, 1]^T \in \mathbb{R}^3$. The motion of the camera frame with respect to the reference frame is sometimes referred to as the *camera pose* and is denoted as $[R, T] \in SE(3)$ with $R \in SO(3)$ being the rotation ($RR^T = I$) and $T \in \mathbb{R}^3$ being the translation.[1]

[1]By setting the motion in $SE(3)$, we only consider rotations and translations. Reflections are not included since it

Therefore, for a point with coordinate $\boldsymbol{X}$ in the reference frame, its image is obtained by

$$\lambda \boldsymbol{x} = [R, T]\, \boldsymbol{X} \in \mathbb{R}^3, \tag{1}$$

where $\lambda > 0$ is the depth of the 3-D point with respect to the camera center. This is the *perspective projection* model of image formation.

The hat operator. One notation that will be extensively used in this chapter is the hat operator "$\widehat{\ }$" that denotes the skew-symmetric matrix associated to a vector in $\mathbb{R}^3$. More specifically, for a vector $\boldsymbol{u} = [u_1, u_2, u_3]^T \in \mathbb{R}^3$, we define

$$\widehat{\boldsymbol{u}} \doteq \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3}, \quad \text{such that} \quad \widehat{\boldsymbol{u}}\boldsymbol{v} = \boldsymbol{u} \times \boldsymbol{v}, \quad \forall \boldsymbol{v} \in \mathbb{R}^3.$$

In particular, $\widehat{\boldsymbol{u}}\boldsymbol{u} = 0 \in \mathbb{R}^3$.

Similarity. We use "$\sim$" to denote similarity. For any pair of vectors or matrices $\boldsymbol{x}$ and $\boldsymbol{y}$ of the same dimension, $\boldsymbol{x} \sim \boldsymbol{y}$ means $\boldsymbol{x} = \alpha \boldsymbol{y}$ for some (nonzero) scalar $\alpha \in \mathbb{R}$.

1.2 3-D reconstruction pipeline

Before we delve into the geometry, we first need to know how the algorithms to be developed can be used. Reconstruction from multiple images often consists of three steps: *feature extraction*, *feature matching*, and *reconstruction using multiple-view geometry*.

Features, or image primitives, are the conspicuous image entities such as corner points, line segments, or structures. The most commonly used image features are points and line segments.

is in $E(3)$, but not in $SE(3)$.

Algorithms for extracting these features can be found in most image processing papers and handbooks [22, 4, 18, 40]. At the end of this chapter we also give an example of using (symmetric) structures as image features for reconstruction.

Feature matching is to establish correspondence of features across different views, which is usually a difficult task. Many techniques have been developed to match features. For instance, when the motion (baseline) between adjacent views is small, feature matching is often called *feature tracking* and typically involves finding an affine transformation between the image patches around the feature points to be matched [34, 54]. Matching across large motion is a much more challenging task and is still an active research area. If a large number of image points are available for matching, some statistical technique such as the RANSAC type of algorithms [13] can be applied. Readers can refer to [23, 40] for details.

Given image features and their correspondences, the camera pose and the 3-D structure of these features can then be recovered using the methods that will be introduced in the rest of this chapter.

1.3 Further readings

Camera calibration. In reality, there are at least two major differences between our camera model and the real camera. First, the focal length f of the real camera is not 1. Second, the origin of the image coordinate frame is usually chosen at the top-left corner of the image instead of at the center of the image. Therefore, we need to map the real image coordinate to our homogeneous representation. This process is called *camera calibration*. In practice, camera calibration is more complicated due to the pixel aspect ratio and nonlinear radial distortion of the image. The simplest calibration scheme is to consider only the focal length and location of the image center. So the

actual image coordinates of a point are given by

$$\lambda \boldsymbol{x} = K\,[R, T]\,\boldsymbol{X}, \quad K = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}, \tag{2}$$

where K is the *calibration matrix* with f being the real focal length and $[x_0, y_0]^T$ being the location of the image center in the image frame. The related theory and algorithms for camera calibration have been studied extensively in the computer vision literature, readers can refer to [41, 71, 60, 5, 20]. In the rest of this chapter, unless otherwise stated, we assume the camera is always calibrated, and we will use equation (1) as the camera model.

Different image surfaces. In the pinhole camera model (Figure 1) we assume that the image plane is a planar surface. However, there are other types of image surfaces such as spheres in omni-directional cameras. For different image surfaces, the theory that we are going to introduce in this chapter still holds with only slight modification. Interested readers please refer to [40, 16] for details.

Other types of projections. Besides the perspective projection model in (1), other types of projections have also been adopted in the literature for various practical or analytical purposes. For example, there are *affine projection* for an uncalibrated camera, *orthographic projection* and *weak perspective projection* for far away objects. For a detailed treatment of these projection models, the reader please refer to [59, 11, 23].

2 Two-view geometry

Let us first study the two-view case. German mathematician Erwin Kruppa [32] is among the first who studied this problem. He showed that given five pairs of corresponding points, the camera motion and structure of the scene can be solved up to finite number of solutions [32, 24, 46]. In practice, however, we usually can obtain more than five points, which may significantly reduce the complexity of the solution and increase the accuracy. In 1980, Longuet-Higgins, based on the epipolar constraint – an algebraic constraint governing two images of a point [33], developed an efficient linear algorithm that requires eight pairs of corresponding points. The algorithm has since been refined several times to reach the current standard *eight-point linear algorithm* [29, 40]. Several variations to this algorithm for coplanar point features and for continuous motions have also been developed [67, 40]. In the remainder of this section, we introduce the epipolar constraint and the eight-point linear algorithm.

2.1 Epipolar constraint and essential matrix

The key issue in solving two-view SFM problem is to identify the algebraic relationship between corresponding image points and the camera motion. Figure 2 shows the relationship between the two camera centers o_1, o_2, the 3-D point p with coordinate $\boldsymbol{X} \in \mathbb{R}^4$, and its two images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Obviously, the three points o_1, o_2 and p form a plane, which implies that the vectors T, $R\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are coplanar. Mathematically, it is equivalent to the triple product of T, $R\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ being zero, i.e.

$$\boldsymbol{x}_2^T \widehat{T} R \boldsymbol{x}_1 = 0. \tag{3}$$

This relationship is called *epipolar constraint* on the pair of images and the camera motion. We denote $E = \widehat{T}R$ and call E the *essential matrix*.

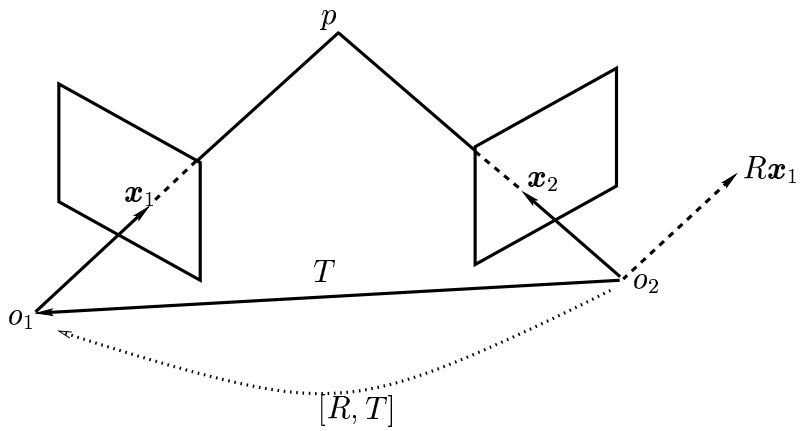

Figure 2: Two views of a point p. The vectors T, $\boldsymbol{x}_2$ and $R\boldsymbol{x}_1$ are the three vectors all expressed in the second camera frame and are coplanar.

2.2 Eight-point linear algorithm

Given image correspondences for $n(\geq 8)$ 3-D points in general positions, the camera motion can be solved linearly. Conceptually it comprises of two steps: First recover the matrix E using n epipolar constraints; then decompose E to obtain motion R and T. However, due to the presence of noise, the recovered matrix E may not be an essential matrix. An additional step of projecting E into the space of essential matrices is necessary prior to the decomposition.

First let use see how to recover E using the epipolar constraint. Denote $E = \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{bmatrix}$ and let $E^s = [e_1, e_4, e_7, e_2, e_5, e_8, e_3, e_6, e_9]^T$ be a *"stacked"* version of E. The epipolar constraint in (3) can be written as

$$(\boldsymbol{x}_1 \otimes \boldsymbol{x}_2)^T E^s = 0, \tag{4}$$

where $\otimes$ denotes the *Kronecker product* of two vectors such that

$$\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 = [x_1x_2, x_1y_2, x_1z_2, y_1x_2, y_1y_2, y_1z_2, z_1x_2, z_1y_2, z_1z_2]^T,$$

given $\boldsymbol{x}_i = [x_i, y_i, z_i]^T$ $(i = 1, 2)$. Therefore, in the absence of noise, given $n(\geq 8)$ pairs of image correspondences $\boldsymbol{x}_1^j$ and $\boldsymbol{x}_2^j$ $(j = 1, 2, \ldots, n)$, we can linearly solve E^s up to a scale using the following equation

$$LE^s \doteq \begin{bmatrix} (\boldsymbol{x}_1^1 \otimes \boldsymbol{x}_2^1)^T \\ (\boldsymbol{x}_1^2 \otimes \boldsymbol{x}_2^2)^T \\ \vdots \\ (\boldsymbol{x}_1^n \otimes \boldsymbol{x}_2^n)^T \end{bmatrix} E^s = 0, \quad L \in \mathbb{R}^{n\times 9}, \tag{5}$$

and choosing E^s as the eigenvector of L^TL associated with the eigenvalue 0.[2] E can then be obtained by "unstacking" E^s.

After obtaining E, we can project it to the space of essential matrix and decompose it to extract the motion, which is summarized in the following algorithm [40]:

Algorithm 1 (Eight-point structure from motion algorithm) *Given $n(\geq 8)$ pairs of image correspondence of points $\boldsymbol{x}_1^j$ and $\boldsymbol{x}_2^j$ $(j = 1, 2, \ldots, n)$, this algorithm recovers the motion $[R, T]$ of the camera (with the first camera frame being the reference frame) in three steps.*

1. **Compute a first approximation of the essential matrix.** *Construct the matrix $L \in \mathbb{R}^{n\times 9}$ as in (5). Choose E^s to be the eigenvector of L^TL associated to its smallest eigenvalue: compute the SVD of $L = U_L S_L V_L^T$ and choose E^s to be the 9^{th} column of V_L. Unstack E^s to obtain the 3×3 matrix E.*

[2] It can be shown that for $n(\leq 8)$ points in general positions, L^TL has only one zero eigenvalue.

2. **Project E onto the space of essential matrices.** *Perform SVD on E such that*

$$E = U diag\{\sigma_1, \sigma_2, \sigma_3\} V^T,$$

where $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$ and $U, V \in SO(3)$. The projection onto the space of essential matrices is $U\Sigma V^T$ with $\Sigma = diag\{1, 1, 0\}$.

3. **Recover motion from by decomposing the essential matrix.** *The motion R and T of the camera can be extracted from the essential matrix using U and V such that*

$$R = UR_z^T\left(\pm\frac{\pi}{2}\right)V^T, \quad \widehat{T} = UR_z\left(\pm\frac{\pi}{2}\right)\Sigma U^T,$$

where $R_z(\alpha)$ means rotation around z-axis by α counterclockwise.

The mathematical derivation and justification for the above projection and decomposition steps can be found in [40].

The above eight-point algorithm in general gives rise to four solutions of motion $[R, T]$. However, only one of them guarantees that the depths of all the 3-D points reconstructed are positive with respect to both camera frames [40]. Therefore, by checking the depths of all the points, the unique physically possible solution can be obtained. Also notice that T is recovered up to a scale. Without any additional scene knowledge, this scale cannot be determined and is often fixed by setting $\| T \| = 1$.

Given the motion $[R, T]$, the next thing is to recover the structure. For point features, that means to recover its depth with respect to the camera frame. For the jth point with depth λ_i^j in the ith ($i = 1, 2$) camera frame, from the fact $\lambda_2^j \boldsymbol{x}_2^j = \lambda_1^j R\boldsymbol{x}_1^j + T$, we have

$$M^j \vec{\lambda^j} \doteq \begin{bmatrix} \widehat{\boldsymbol{x}_2}^j R\boldsymbol{x}_1^j & \widehat{\boldsymbol{x}_2}^j T \end{bmatrix} \begin{bmatrix} \lambda_1^j \\ 1 \end{bmatrix} = 0. \tag{6}$$

So λ_1^j can be solved by finding the eigenvector of ${M^j}^T M^j$ associated to its smallest eigenvalue.

Example 2 *As shown in Figure 3, two images of a calibration cube are present. Twenty-three pairs of feature points are extracted using Harris corner detector , and the correspondence are established manually. The reconstruction is performed using the eight-point algorithm and depth calculation. The three angles for an orthogonal corner are close to right angles. The coplanarity of*

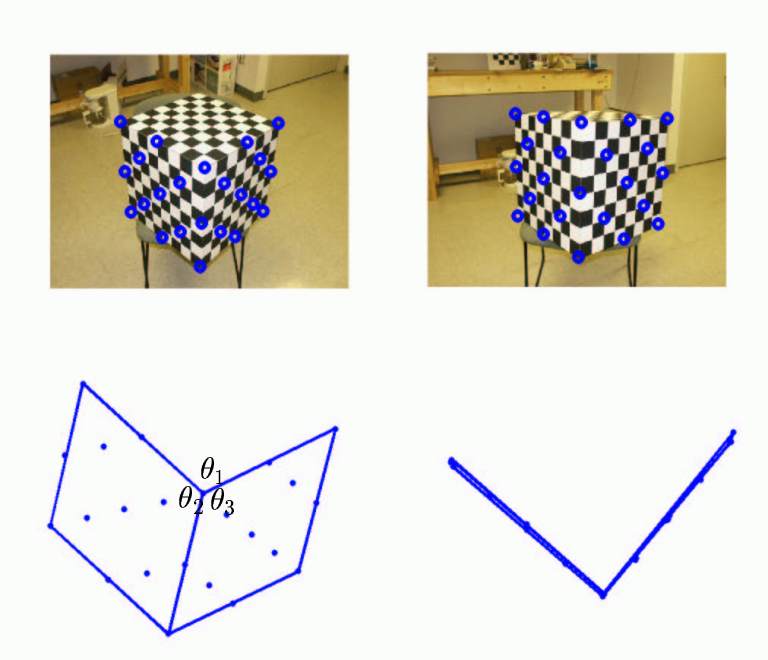

Figure 3: The two images of a calibration cube and two views of the reconstructed structure. The three angles are $\theta_1 = 89.7°$, $\theta_2 = 92.3°$ and $\theta_3 = 91.9°$.

points in each plane are almost preserved. Overall, the structure is reconstructed fairly accurate.

Coplanar features and homography. Up to now, we assume that all the 3-D points are in general positions. In practice, it is not unusual that all the points resides on the same 3-D plane, i.e. they are *coplanar*. The eight-point algorithm will fail due to the fact that the matrix $L^T L$ (see (5)) will have more than one eigenvalues being zero. Fortunately, besides epipolar constraint, there exists

another constraint for coplanar points. This is the *homography* between the two images, which can be described using a matrix $H \in \mathbb{R}^{3\times 3}$:

$$H = R + \frac{1}{d} T N^T \tag{7}$$

with $d > 0$ denoting the distance from the plane to the first camera center and N being the unit normal vector of the plane expressed in the first camera frame with $\lambda_1 N^T \boldsymbol{x}_1 + d = 0$.[3] It can be shown that the two images $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ of the same point are related by:

$$\widehat{\boldsymbol{x}_2} H \boldsymbol{x}_1 = 0. \tag{8}$$

Using the homography relationship, we can recover the motion from two views with a similar procedure to the epipolar constraints: First H can also be calculated linearly using $n(\geq 4)$ pairs of corresponding image points. The reason that the minimum number of point correspondences is four instead of eight is that each pair of image points provide two independent equations on H through (8). Then the motion $[R, T]$ as well as the plane normal vector N can be obtained by decomposing H. However, the solution for the decomposition is more complicated. This algorithm is called *four-point algorithm* for coplanar features. Interested readers please refer to [67, 40].

2.3 Further Readings

The eight-point algorithm introduced in this section is for general situations. In practice, however, there are several caveats:

[3]The homography is not limited to two image points in two camera frames, it is for the coordinates of the 3-D point on the plane expressed in any two frames (with one frame being the reference frame). In particular, if the second frame is chosen with its origin lying on the plane, then we have a homography between the camera and the plane.

Small baseline motion and continuous motion. If $\| T \|$ is small and data is noisy, the reconstruction algorithm often would fail. This is the *small baseline* case. Readers can refer to [40, 65] for special algorithm dealing with this situation. When the baseline become infinitesimally small, we have the case of *continuous motion*, for which the algebra becomes somewhat different from the discrete case. For a detailed analysis and algorithm, please refer to [39, 40].

Multiple-body motions. For the case in which there are multiple moving objects in the scene, there exists a more complicated *multiple-body epipolar constraint*. The reconstruction algorithm can be found in [66, 40].

Uncalibrated camera. If the camera is uncalibrated, the essential matrix E in the epipolar constraint should be substituted by the *fundamental matrix* F with $F = K^{-T}EK^{-1}$, where $K \in \mathbb{R}^{3\times 3}$ is the calibration matrix of the camera, defined in equation (2). The analysis and affine reconstruction for the uncalibrated camera can be found in [23, 40].

Critical surface. There are certain degenerate positions of the points for which the reconstruction algorithm would fail. These configurations are called *critical surfaces* for the points. Detailed analysis are available in [44, 40].

Numerical problems and optimization. To obtain accurate reconstruction, some numerical issues such data normalization need to be addressed before applying the algorithm. These are discussed in [40]. Notice that the eight-point algorithm is only a sub-optimal algorithm; various nonlinear "optimal" algorithms have been designed, which can be found in [64, 40].

3 Multiple-view geometry

In this section, we study the case for reconstruction from more than two views. Specifically, we present a set of *rank conditions* on a *multiple-view matrix* [37, 38, 27]. The epipolar constraint for two views is just a special case implied by the rank condition. As we will see, the multiple-view matrix associated to a geometric entity (point, line or plane) is exactly the 3-D information that is missing in a single 2-D image but encoded in multiple ones. This approach is compact in representation, intuitive in geometry, and simple in computation. Moreover, it provides a unified framework for describing multiple views of all types of features and incidence relations in 3-D space [40].

3.1 Rank condition on multiple views of point feature

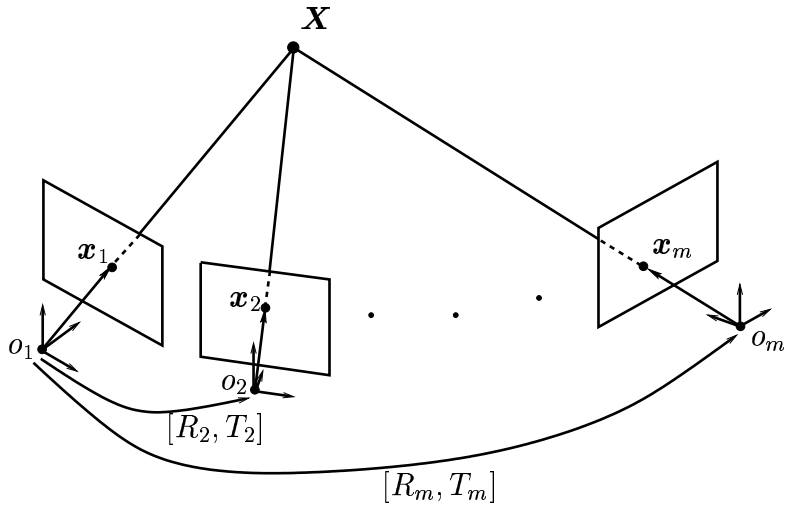

Figure 4: Multiple images of a 3-D point $\boldsymbol{X}$ in m camera frames.

First, let us look at the case of multiple images of a point. As shown in Figure 4, multiple

images $\boldsymbol{x}_i$ of a 3-D point $\boldsymbol{X}$ with camera motions $[R_i, T_i]$ ($i = 1, 2, \ldots, m$) satisfy

$$\lambda_i \boldsymbol{x}_i = [R_i, T_i]\, \boldsymbol{X}, \tag{9}$$

where λ_i is the point depth in the ith camera frame. Multiplying $\widehat{\boldsymbol{x}}_i$ on both sides of (9), we have

$$[\widehat{\boldsymbol{x}}_i R_i \quad \widehat{\boldsymbol{x}}_i T_i]\, \boldsymbol{X} = 0. \tag{10}$$

Without loss of generality, we choose the first camera frame as the reference frame such that $R_1 = I, T_1 = 0$, and $\boldsymbol{X} = \left[\begin{smallmatrix} \lambda_1 \boldsymbol{x}_1 \\ 1 \end{smallmatrix}\right]$. Therefore, for $i = 2, 3, \ldots, m$, (10) can be transformed into

$$[\widehat{\boldsymbol{x}}_i R_i \boldsymbol{x}_1 \quad \widehat{\boldsymbol{x}}_i T_i] \begin{bmatrix} \lambda_1 \\ 1 \end{bmatrix} = 0. \tag{11}$$

Stacking the left side of the above equations for all $i = 2, \ldots, m$, we have

$$M \begin{bmatrix} \lambda_1 \\ 1 \end{bmatrix} \doteq \begin{bmatrix} \widehat{\boldsymbol{x}}_2 R_2 \boldsymbol{x}_1 & \widehat{\boldsymbol{x}}_2 T_2 \\ \widehat{\boldsymbol{x}}_3 R_3 \boldsymbol{x}_1 & \widehat{\boldsymbol{x}}_3 T_3 \\ \vdots & \vdots \\ \widehat{\boldsymbol{x}}_m R_m \boldsymbol{x}_1 & \widehat{\boldsymbol{x}}_m T_m \end{bmatrix} \begin{bmatrix} \lambda_1 \\ 1 \end{bmatrix} = 0 \quad \in \mathbb{R}^{3(m-1)}. \tag{12}$$

The matrix $M \in \mathbb{R}^{3(m-1)\times 2}$ is called the *multiple-view matrix* for point features. The above relationship is summarized in the following theorem:

Theorem 3 (Rank condition on multiple-view matrix for point features) *The multiple-view matrix M satisfies the following rank conditions:*

$$\mathit{rank}(M) \leq 1. \tag{13}$$

Furthermore, $M \left[\begin{smallmatrix} \lambda_1 \\ 1 \end{smallmatrix}\right] = 0$ with λ_1 being the depth of the point in the first camera frame.

The detailed proof of the theorem can be found in [37, 40].

The implication of the above theorem is multi-fold. First, the rank of M can be used to detect the configuration of the cameras and the 3-D point. If rank$(M) = 1$, the cameras and the 3-D point are in the general positions; and the location of the point can be determined (up to a scale) by triangulation. If rank$(M) = 0$, then all the camera centers and the point are collinear; and the point can only be determined up to a line. Second, all the algebraic constraints implied by the rank condition involve no more than three views, which means that for point features a fourth image no longer imposes any new algebraic constraint. Last but not the least, the rank condition on M uses all the data simultaneously, which significantly simplifies the calculation. Also note that the rank condition on the multiple-view matrix implies the epipolar constraint. For the ith ($i > 1$) view and the first view, the fact that $\widehat{\boldsymbol{x}}_i R_i \boldsymbol{x}_1$ and $\widehat{\boldsymbol{x}}_i T_i$ are linearly dependent is equivalent to the epipolar constraint between the two views.

3.2 Linear reconstruction algorithm

Now we demonstrate how to apply the multiple-view rank condition in reconstruction. Specifically, we show the linear reconstruction algorithm for point features. Given $m(\geq 3)$ images of $n(\geq 8)$ points in 3-D space with $\boldsymbol{x}_i^j$ ($i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$), the structure (depths λ_i^1 with respect to the first camera frame) and the camera motions $[R_i, T_i]$ can be recovered in two major steps.

First for the jth point, a multiple-view matrix M^j associated with its images satisfies

$$M^j \begin{bmatrix} 1 \\ \frac{1}{\lambda_1^j} \end{bmatrix} = \begin{bmatrix} \widehat{\boldsymbol{x}_2}^j R_2 \boldsymbol{x}_1^j & \widehat{\boldsymbol{x}_2}^j T_2 \\ \widehat{\boldsymbol{x}_3}^j R_3 \boldsymbol{x}_1^j & \widehat{\boldsymbol{x}_3}^j T_3 \\ \vdots & \vdots \\ \widehat{\boldsymbol{x}_m}^j R_m \boldsymbol{x}_1^j & \widehat{\boldsymbol{x}_m}^j T_m \end{bmatrix} \begin{bmatrix} 1 \\ \frac{1}{\lambda_1^j} \end{bmatrix} = 0. \tag{14}$$

The above equation implies that if the set of motions $g_i = [R_i, T_i]$'s are known for all $i = 2, \ldots, m$, the depth λ_1^j of the jth point with respect to the first camera frame can be recovered by computing the kernel of M^j. We denote $\alpha^j = \frac{1}{\lambda_1^j}$.

Similarly, for the ith image ($i = 2, \ldots, m$), if α^j's are known for all $j = 1, 2, \ldots, n$, the estimation of $R_i = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ and T_i is equivalent to solving the stacked vectors

$$R_i^s = [r_{11}, r_{21}, r_{31}, r_{12}, r_{22}, r_{33}, r_{13}, r_{23}, r_{33},]^T \in \mathbb{R}^9$$

and $T_i \in \mathbb{R}^3$ using the equation

$$P_i \begin{bmatrix} R_i^s \\ T_i \end{bmatrix} \doteq \begin{bmatrix} \widehat{\boldsymbol{x}}_i^{\,1} \otimes \boldsymbol{x}_1^1 & \alpha^1 \widehat{x}_i^{\,1} \\ \widehat{\boldsymbol{x}}_i^{\,2} \otimes \boldsymbol{x}_1^2 & \alpha^2 \widehat{x}_i^{\,2} \\ \vdots & \vdots \\ \widehat{\boldsymbol{x}}_i^{\,n} \otimes \boldsymbol{x}_1^n & \alpha^n \widehat{x}_i^{\,n} \end{bmatrix} \begin{bmatrix} R_i^s \\ T_i \end{bmatrix} = 0 \in \mathbb{R}^{3n}, \quad P_i \in \mathbb{R}^{3n \times 12}, \tag{15}$$

where $\otimes$ is the *Kronecker product* between two matrices. It can be shown that P_i has rank 11 if $n \geq 6$ points in general positions are present. Therefore the solution of $[R_i, T_i]$ will be unique up to a scale for $n \geq 6$.

Obviously, if no noise is present on the images, the recovered motion and structure will be the same with what can be recovered from the two-view eight-point algorithm. However, in the

presence of noise, it is desired to use the data from all the images. In order to use the data simultaneously, a reasonable approach is to iterate between reconstructions of motion and structure, i.e. initializing the structure or motions, then alternating between (14) and (15) until the structure and motion converge.

Motions $[R_i, T_i]$ can then be estimated up to a scale by performing SVD on P_i as in (15).[4] Denote $\tilde{R}_i$ and $\tilde{T}_i$ to be the estimates from the eigenvector of $P_i^T P_i$ associated to the smallest eigenvalue. Let $\tilde{R}_i = U_i S_i V_i^T$ be the SVD of $\tilde{R}_i$. Then $R_i \in SO(3)$ and T_i are given by

$$R_i = \text{sign}(\det(U_i V_i^T)) U_i V_i^T \in SO(3), \quad \text{and} \quad T_i = \frac{\text{sign}(\det(U_i V_i^T))}{(\det(S_i))^{\frac{1}{3}}} \in \mathbb{R}^3. \tag{16}$$

In this algorithm, the initialization can be done using the eight-point algorithm for two views. The initial estimate on the motion of second frame $[R_2, T_2]$ can be obtained using the standard two-view eight-point algorithm. Initial estimates of the point depth is then

$$\alpha^j = -\frac{(\widehat{\boldsymbol{x}}_2^j T_2)^T \widehat{\boldsymbol{x}}_2^j R_2 \boldsymbol{x}_1^j}{\| \widehat{\boldsymbol{x}}_2^j T_2 \|^2}, \quad j = 1, \ldots, n. \tag{17}$$

In the multiple-view case, the least square estimates of point depths $\alpha^j = \frac{1}{\lambda_1^j}$, $j = 1, \ldots, n$ can be obtained from (14) as

$$\alpha^j = -\frac{\sum_{i=2}^m (\widehat{\boldsymbol{x}}_i^j T_i)^T \widehat{\boldsymbol{x}}_i^j R_i \boldsymbol{x}_1^j}{\sum_{i=2}^m \| \widehat{\boldsymbol{x}}_i^j T_i \|^2}, \quad j = 1, \ldots, n. \tag{18}$$

By iterating between the motion estimation and structure estimation, we expect that the estimates on structure and motion converge. The convergence criteria may vary for different situations. In practice we choose the reprojected images error as the convergence criteria. For the jth 3-D point, the estimate of its 3-D location can be obtained as $\lambda_1^j \boldsymbol{x}_1^j$ and the reprojection on the ith

[4] Now we assume that the cameras are all calibrated, which is the case of Euclidean reconstruction. This algorithm also works for uncalibrated case.

image is obtained $\tilde{\boldsymbol{x}}_i{}^j \sim \lambda_1^j R_i \boldsymbol{x}_1^j + T_i$. Its reprojection error is then $\sum_{i=2}^m \parallel \boldsymbol{x}_i^j - \tilde{\boldsymbol{x}}_i{}^j \parallel^2$. So the algorithm keep iterating until the summation of reprojection errors over all points are below some threshold ϵ.

The algorithm is summarized below:

Algorithm 4 (A factorization algorithm for multiple-view reconstruction) *Given $m(\geq 3)$ images $\boldsymbol{x}_1^j, \boldsymbol{x}_2^j, \ldots, \boldsymbol{x}_m^j$, $j = 1, 2, \ldots, n$ of $n(\geq 8)$ points, the motions $[R_i, T_i]$, $i = 2, \ldots, m$ and the structure of the points with respect to the first camera frame α^j, $j = 1, 2, \ldots, n$ can be recovered as follows:*

1. *Set the counter $k = 0$. Compute $[R_2, T_2]$ using the eight-point algorithm, then get an initial estimate of α^j from (17) for each $j = 1, 2, \ldots, n$. Normalize $\alpha^j \leftarrow \alpha^j/\alpha^1$ for $j = 1, 2, \ldots, n$.*

2. *Compute $\left[\tilde{R}_i, \tilde{T}_i\right]$ from the eigenvector of $P_i^T P_i$ corresponding its smallest eigenvalue for $i = 2, \ldots, m$.*

3. *Compute $[R_i, T_i]$ from (16) for $i = 2, 3, \ldots, m$.*

4. *Compute α_{k+1}^j using (18) for each $j = 1, 2, \ldots, n$. Normalize so that $\alpha^j \leftarrow \alpha^j/\alpha^1$ and $T_i \leftarrow \alpha_{k+1}^1 T_i$. Use the newly recovered α^j's and motion $[R_i, T_i]$'s to computer the reprojected image $\tilde{\boldsymbol{x}}$ for each point in all views.*

5. *If the reprojection error $\sum \parallel \boldsymbol{x} - \tilde{\boldsymbol{x}} \parallel^2 > \epsilon$ for some threshold $\epsilon > 0$, then $k \leftarrow k + 1$ and go to step 2, otherwise stop.*

The above algorithm is a direct derivation from the rank condition. There are techniques to improve its numerical stability and statistical robustness for specific situations [40].

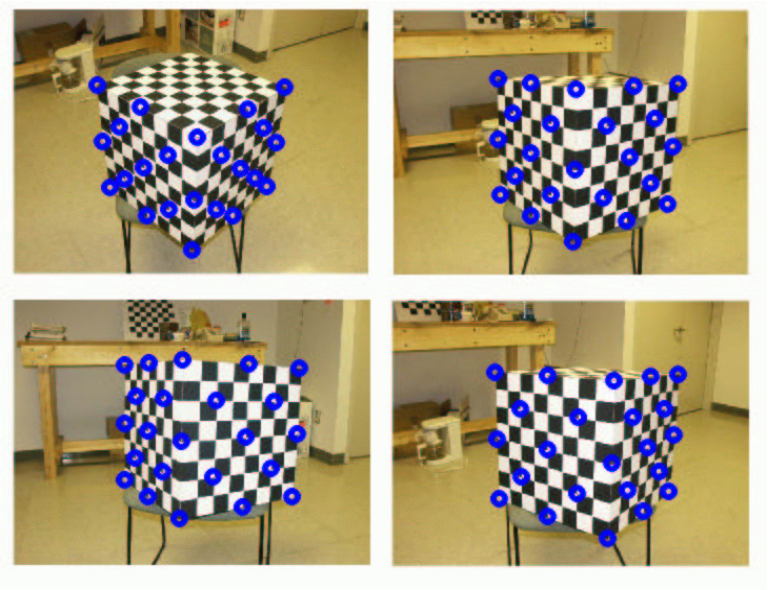

Figure 5: The four images used to reconstruct the calibration cube.

Example 5 *The algorithm proposed in previous section has been tested extensively in both simulation [37] and experiments [40]. Figure 5 shows the four images used to reconstruct the calibration cube. The points are marked in circles. Two views of the reconstruction results are shown in Figure 6. The recovered angles are more accurate than the results from Figure 3. Visually, the coplanarity of the points is preserved well.*

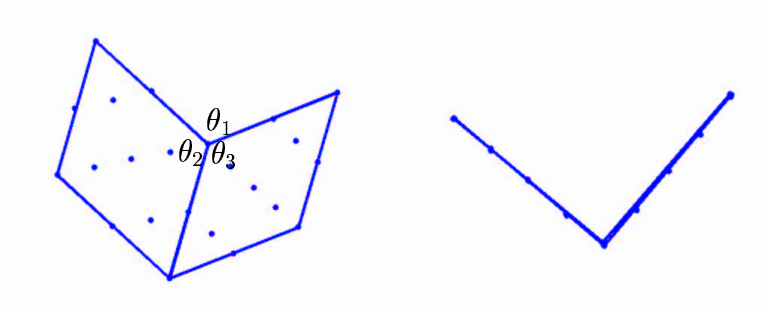

Figure 6: The two views of the reconstructed structure. The three angles are $\theta_1 = 89.9°$, $\theta_2 = 91.0°$, and $\theta_3 = 90.6°$.

Coplanar features. The treatment for coplanar point features is similar to the general case. As-

sume the equation of the plane in the first camera frame is $\left[\pi_1^T, \pi_2\right] \boldsymbol{X} = 0$ with $\pi_1 \in \mathbb{R}^3$ and $\pi_2 \in \mathbb{R}$. Simply appending the 1×2 block $\left[\pi_1^T \boldsymbol{x}_1, \pi_2\right]$ to the end of M in (12), then the rank condition in Theorem 3 still holds. Since $\pi_1 = N$ is the unit normal vector of the plane and $\pi_2 = d$ is the distance from the first camera center to the plane, the rank condition implies $d(\widehat{\boldsymbol{x}}_i R_i \boldsymbol{x}_1) - (N^T \boldsymbol{x}_1)(\widehat{\boldsymbol{x}}_i T_i) = 0$, which is obviously equivalent to the homography between the ith and the first views (see (8)). As for reconstruction, we can use the four-point algorithm to initialize the estimation of the homography and then perform similar iteration scheme to obtain motion and structure. The algorithm can be found in [36, 51].

3.3 Further readings

Multilinear constraints and factorization algorithm. There are two other approaches dealing with multiple-view reconstruction. The first approach is to use the so-called *multilinear constraints* on multiple images of a 3-D point or line. For small number of views, these constraints can be described in terms of tensorial notations [52, 53, 23]. For example, the constraints for $m = 3$ can be described using *trifocal tensors*. For large number of views ($m \geq 5$), the tensor is difficult to describe. The reconstruction is then to calculate the trifocal tensors first and factorize the tensors for camera motions [2]. An apparent disadvantage is that it is hard to choose the right "three-view sets" and also difficult to combine the results. Another approach is to apply some factorization scheme to iteratively estimate the structure and motion [57, 23, 40]. The problem for this approach is that it is hard to initialize.

Universal multiple-view matrix and rank conditions. The reconstruction algorithm in this section was only for point features. Algorithms have also been designed for lines features [56, 35].

In fact the multiple-view rank condition approach can be extended to all different types of features such as line, plane, mixed line and point and even curves. This leads to a set of rank conditions on a *universal multiple-view matrix*. For details please refer to [38, 40].

Dynamical scenes. In the constraint we developed in this section is for static scene. If the scene is *dynamic*, i.e. there are moving objects in the scene, a similar type of rank condition can be obtained. This rank condition is obtained by incorporating the dynamics of the objects in 3-D space into its own description and lift the 3-D moving points into a higher dimensional space in which it is static. For details please refer to [27, 40].

Orthographic projection. Finally, note that the linear algorithm and the rank condition are for the perspective projection model. If the scene is far from the camera, then the image can be modeled using orthographic projection, and the Tomasi-Kanade factorization method can be applied [59]. Similar factorization algorithm for other types of projections and dynamics have also been developed [48, 49, 8, 21].

4 Utilizing prior knowledge of the scene – symmetry

In this section we study how to incorporate scene knowledge into the reconstruction process. In our daily life, especially in a man-made environment, there exist all types of "regularity." For objects, regular shapes such as rectangle, square, diamond, and circle always attract our attention. For spatial relationship between objects, orthogonality, parallelism, and similarity are the conspicuous ones. Interestingly, all the above regularities can be described using the notion of *symmetry*. For instance, a rectangular window has one rotational symmetry and two reflective symmetry; the same windows on the same wall have translational symmetry; the corner of a cube displays rotational

symmetry.

4.1 Symmetric multiple-view rank condition

There are many studies using instances of symmetry in the scene for reconstruction purposes [25, 19, 1, 6, 73, 74, 70, 3, 28]. Recently, a set of algorithms using symmetry for reconstruction from a single image have been developed [25]. The main idea is to use the so-called *equivalent images* encoded in a single image of a symmetric object. Figure 7 illustrates this notion for the case of a reflective symmetry. We attach an *object coordinate frame* to the symmetric object and set it as the reference frame. 3-D points $\boldsymbol{X}$ and $\boldsymbol{X}'$ are related by some symmetric transformation g (in the object frame) with $g(\boldsymbol{X}) = \boldsymbol{X}'$. In the image obtained from viewpoint O, then the image of $\boldsymbol{X}'$ can be interpreted as the image of $\boldsymbol{X}$ viewed from the virtual viewpoint O' that is the correspondence of O under the same symmetry transformation g. The image of $\boldsymbol{X}'$ is an called an *equivalent image* of $\boldsymbol{X}$ viewed from O'. Therefore, given a single image of a symmetric object, we have multiple equivalent images of this object. The number of all equivalent images is the number of all symmetry of the object.

In modern mathematics, symmetry of an object are characterized by a *symmetry group* with each element in the group representing a transformation under which the object is invariant [68, 25]. For example, a rectangle possesses two reflective symmetry and one rotational symmetry. We can use the group theoretic notation to define 3-D symmetric object as in [25].

Definition 6 *Let S be a set of 3-D points. It is called a* symmetric structure *if there exists an non-trivial subgroup G of the Euclidean group $E(3)$ acting on S such that for any $g \in G$, g defines an isomorphism from S to itself. G is called the* symmetry group *of S.*

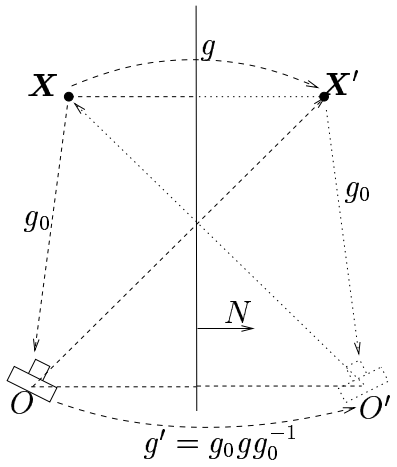

Figure 7: $\boldsymbol{X}$ and $\boldsymbol{X}'$ are corresponding points under reflective symmetry transformation g (expressed in the object frame) such that $\boldsymbol{X}' = g(\boldsymbol{X})$. N is the normal vector of the mirror plane. The motion between the camera frame and the object frame is g_0. Hence the image of $\boldsymbol{X}'$ in real camera can be viewed as the image of $\boldsymbol{X}$ viewed by a virtual camera with pose $g_0 g$ with respect to the object frame or $g_0 g g_0^{-1}$ with respect to the real camera frame.

Under this definition, the possible symmetry on an 3-D object are reflective symmetry, translational symmetry, rotational symmetry, and any combination of them. For any point $p \in S$ on the object, its symmetric correspondence for $g \in G$ is $g(p) \in S$. The images of p and $g(p)$ are denoted as $\boldsymbol{x}$ and $g(\boldsymbol{x})$.

Let the symmetry transformation in the object frame be $g = [R, T] \in G$ ($R \in O(3)$ and $T \in \mathbb{R}^3$). As illustrated in Figure 7, if the transformation from the object (reference) frame to the real camera frame is $g_0 = [R_0, T_0]$, the transformation from the reference frame to the virtual camera frame is $g_0 g$. Furthermore, the transformation from the real camera frame to the virtual camera frame is

$$g' = g_0 g g_0^{-1} = [R', T'] = \left[R_0 R R_0^T, (I - R_0 R R_0^T) T_0 + R_0 T\right]. \tag{19}$$

For the symmetric object S, assume its symmetry group G has m elements $g_i = [R_i, T_i]$,

$i = 1, 2, \ldots, m$. Then the transformation between the ith virtual camera and the real camera is $g_i' = [R_i', T_i']$ ($i = 1, 2, \ldots, m$) as can be calculated from (19). Given any point $\boldsymbol{X} \in S$ with image $\boldsymbol{x}$ and its equivalent images $g_i(\boldsymbol{x})$'s, we can define the *symmetric multiple-view matrix*

$$M(\boldsymbol{x}_i) = \begin{bmatrix} \widehat{g_1(\boldsymbol{x})} R_1' \boldsymbol{x} & \widehat{g_1(\boldsymbol{x})} T_1' \\ \widehat{g_2(\boldsymbol{x})} R_2' \boldsymbol{x} & \widehat{g_2(\boldsymbol{x})} T_2' \\ \vdots & \vdots \\ \widehat{g_m(\boldsymbol{x})} R_m' \boldsymbol{x} & \widehat{g_m(\boldsymbol{x})} T_m' \end{bmatrix}. \tag{20}$$

According to Theorem 3, it satisfies the *symmetric multiple-view rank condition*:

$$\text{rank}\,(M\,(\boldsymbol{x})) \leq 1. \tag{21}$$

4.2 Reconstruction from symmetry

Using the symmetric multiple-view rank condition and a set of n symmetric correspondences (points), we can solve for $g_i' = [R_i', T_i']$ and the structure of the points in S in the object frame using an algorithm similar to Algorithm 4. However, a further step is still necessary to recover $g_0 = [R_0, T_0]$ for the camera pose with respect to the object frame. This can be done by solving the following Lyapunov type of equations:

$$g_i' g_0 - g_0 g_i = 0, \quad \text{or} \quad R_i' R_0 - R_0 R = 0 \quad \text{and} \quad T_0 = (I - R')^\dagger (T' - R_0 T). \tag{22}$$

Since g_i' and g_i are known, g_0 can be solved. The detailed treatment can be found in [25, 40].

As can be seen in the example at the end of this section, symmetry-based reconstruction is very accurate and requires minimum amount of data. A major reason is that the baseline between the real camera and the virtual one is often large due to the symmetry transformation. Therefore

degenerate cases will occur if the camera center is invariant under the symmetry transformation. For example, if the camera lies on the mirror plane for a reflective symmetry, the structure can only be recovered up to some ambiguities [25].

The reconstruction for some special cases can be simplified without explicitly using the symmetric multiple-view rank condition (e.g., see [3]). For the reflective symmetry, the structure with respect to the camera frame can be calculated using only two pairs of symmetric points. Assume the image of two pairs of reflective points are $\boldsymbol{x}$, $\boldsymbol{x}'$ and $\boldsymbol{y}$, $\boldsymbol{y}'$. Then the image line connecting $\boldsymbol{x}$ and $\boldsymbol{x}'$ is obtained by $\boldsymbol{l}_x \sim \widehat{\boldsymbol{x}}\boldsymbol{x}'$. Similarly, $\boldsymbol{l}_y$ connecting $\boldsymbol{y}$ and $\boldsymbol{y}'$ satisfies $\boldsymbol{l}_y \sim \widehat{\boldsymbol{y}}\boldsymbol{y}'$. It can be shown that the unit normal vector N (see Figure 7) of the reflection plane satisfies

$$\begin{bmatrix} \boldsymbol{l}_x^T \\ \boldsymbol{l}_y^T \end{bmatrix} N = 0, \tag{23}$$

from which N can be solved. Assume the depth of $\boldsymbol{x}$ and $\boldsymbol{x}'$ are λ and λ', then they can be calculated using the follow relationship

$$\begin{bmatrix} \widehat{N}\boldsymbol{x} & -\widehat{N}\boldsymbol{x}' \\ N^T\boldsymbol{x} & N^T\boldsymbol{x}' \end{bmatrix} \begin{bmatrix} \lambda \\ \lambda' \end{bmatrix} = \begin{bmatrix} 0 \\ 2d \end{bmatrix}, \tag{24}$$

where d is the distance from the camera center to the reflection plane.

Planar symmetry. Planar symmetric objects are widely present in man-made environment. Regular shapes such as square and rectangle are often good landmarks for mapping and recognition tasks. For a planar symmetric object, its symmetry form a subgroup G of $E(2)$ instead of $E(3)$. Let $g \in E(2)$ be one symmetry of a planar symmetric object. Recall that there exists a homography H_0 between the object frame and camera frame. Also between the original image and the equivalent image generated by g, there exists another homography H. It can be shown that $H = H_0 g H_0^{-1}$. Therefore all the homographies generated from the equivalent images form a *homography group*

$H_0 G H_0^{-1}$, which is conjugate to G [3]. This fact can be used in testing if an image is the image of an object with desired symmetry. Given the image, by calculating the homographies from all equivalent images based on the hypothesis, we can check the group relationship of the homography group and decide if the desired symmetry exists [3]. In case the object is a rectangle, the calculation can be further simplified using the notion of *vanishing point* as illustrated in the following example.

Example 7 (Symmetry-based reconstruction for a rectangular object) *For a rectangle in 3-D space, the two pairs of parallel edges generate two vanishing points $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ in the image. As a 3-D vector, $\boldsymbol{v}_i$ $(i = 1, 2)$ can also be interpreted as the vector from the camera center to the vanishing point, and hence must be parallel to the pair of parallel edges. Therefore, we must have $\boldsymbol{v}_1 \perp \boldsymbol{v}_2$. So by checking the angle between the two vanishing points, we can decide if a region can be the image of a rectangle. Figure 8 demonstrates the reconstruction based on the assumption of a rectangle. The two sides of the cube are two rectangles (in fact squares). The angles between the vanishing points are $89.1°$ and $92.5°$, respectively. The reconstruction is performed using only one image and six points. The angle between the two planes is $89.2°$.*

4.3 Further readings

Symmetry and vision. Symmetry is a strong vision cue in human vision perception and has been extensively discussed in psychology and cognition research [43, 47, 45]. It has been noticed that symmetry is useful for face recognition [62, 63, 61].

Symmetry in statistical context. Besides geometric symmetry that has been discussed in this section, symmetry in the sense of statistics have also been studied and utilized. Actually, the

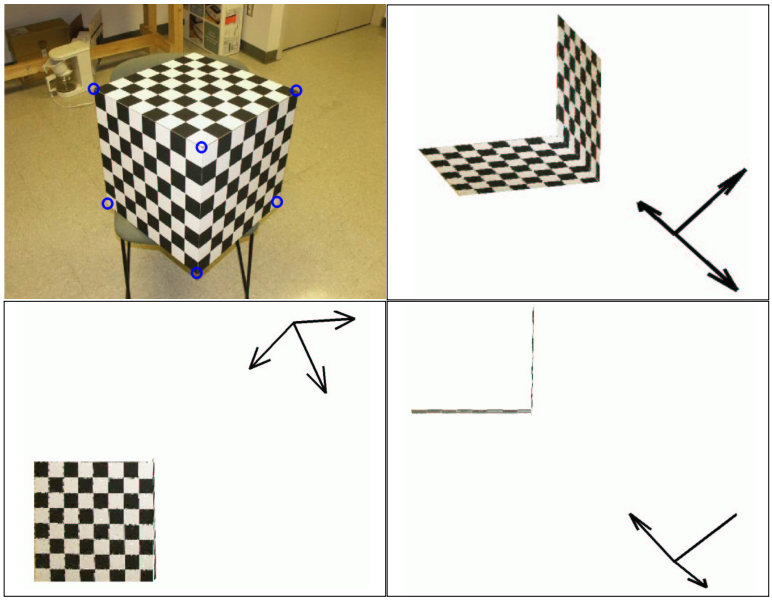

Figure 8: Reconstruction from a single view of the cube with six corner points marked (in blue). An arbitrary view, a side view and a bird view of the reconstructed and rendered rectangles are displayed. The coordinate frame shows the recovered camera pose.

computational advantages of symmetry were first explored in the statistical context, such as the study of isotropic texture [17, 69, 42]. It was the work of [14, 15, 42] that provided a wide range of efficient algorithms for recovering the orientation of a textured plane based on the assumption of isotropy or weak isotropy.

Symmetry of surfaces and curves. While we only utilized symmetric points in this chapter, the symmetry of surfaces has also been exploited. [55, 72] used the surface symmetry for human face reconstruction. [25, 19] studied reconstruction of symmetric curves.

5 Comprehensive examples and experiments

Finally, we discuss several applications of using the techniques introduced in this chapter. These applications all involve reconstruction from a single or multiple images for the purpose of robotic navigation and mapping. While not all of them are fully automatic, they demonstrate the potential of the techniques discussed in this chapter and some future directions for improvement.

5.1 Automatic landing of unmanned aerial vehicles

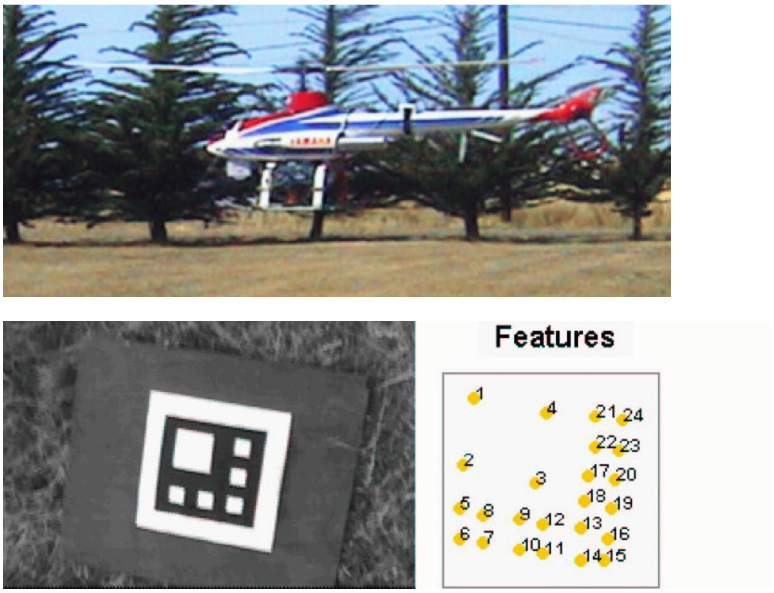

Figure 9: Top: A picture of the UAV in landing process. Bottom Left: An image of the landing pad viewed from an on-board camera. Bottom Right: Extracted corner features from the image of the landing pad. (Photo courtesy of O. Shankeria)

Figure 9 displays the experiment setup for applying the multiple-view rank condition based algorithm in automatic landing of UAV in University of California at Berkeley [51]. In this experiment, the UAV is a model helicopter (Figure 9 Top) with an on-board video camera facing

downwards searching for the landing pad (Figure 9 Bottom left) on the ground. When the landing pad is found, corner points are extracted from each image in the video sequence (Figure 9 Bottom right). The pose and the motion of the camera (and the UAV) are estimated using the feature points in the images. Previously the four-point two-view reconstruction algorithm for planar features was tested. However, its results were noisy (Figure 10 Left). Later an nonlinear two-view algorithm was developed, but it was time consuming. Finally, by adopting the rank condition based multiple-view algorithm, the landing problem was solved.

The multiple-view reconstruction are performed for every four images at 10Hz. The algorithm is a modification of the Algorithm 4 introduced in this chapter, which is specifically designed for coplanar features. Figure 10 shows the comparison of this algorithm with other algorithms and sensors. The multiple-view algorithm is more accurate than both two-view algorithms (Figure 10 Left) and is close to the results obtained from differential GPS and INS sensors (Figure 10 Right). The overall error for this algorithm is less then 5cm in distance and 4° for rotation [51].

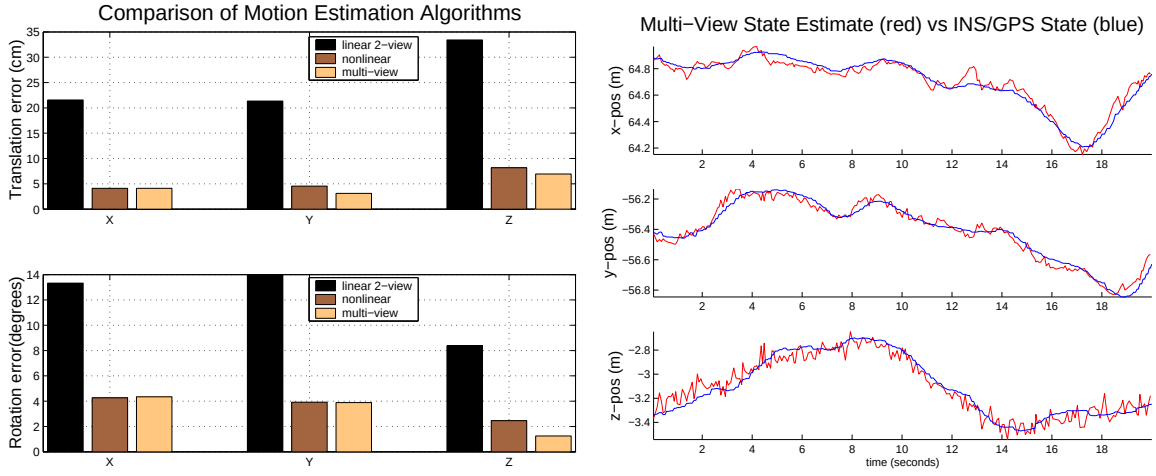

Figure 10: Left: Comparison of the multiple-view rank condition based algorithm with two view linear algorithm and nonlinear algorithm. Right: Comparison of the results from multiple-view rank condition based algorithm (red line) with GPS and INS results (blue line). (Image courtesy of O. Shankeria.)

5.2 Automatic symmetry cell detection, matching and reconstruction

In Section 4, we discussed symmetry-based reconstruction techniques from a single image. Here we present a comprehensive example that performs symmetry-based reconstruction from multiple views [3, 28]. In this example, the image primitives are no longer point or line. Instead we use the *symmetry cells* as features. The example includes three steps:

Feature extraction. By *symmetry cell* we mean a region in the image that is an image of a desired symmetric object in 3-D. In this example, the symmetric object we choose is rectangle. So the symmetry cells are images of rectangles. Detecting symmetry cells (rectangles) includes two steps. First, we perform color-based segmentation on the image. Then, for all the detected four-sided regions, we test if its two vanishing points are perpendicular to each other and decide if it can be an image of a rectangle in 3-D space. Figure 11 demonstrates this for the picture of an indoor scene. In this picture, after color segmentation, all four-sided regions with reasonable sizes are detected and marked with purple boundaries. Then each four-sided polygon is passed through the vanishing point test. For those polygons passing the test, we denote them as symmetry cells and recover the object frames with the symmetry-based algorithm. Each individual symmetry cell is recovered with a different scale. Please note that this test is a hypothesis testing step, it cannot verify if the object in 3-D is actually a desired symmetric object, instead it can confirm that its image satisfies the condition of the test.

Feature matching. For each cell we can calculate its 3-D shape. Then for two cells in two images, by comparing their 3-D shape and colors, we can decide if they are matching candidates. In the case of existence of many similar symmetry cells, we need to use a matching graph [28]. That is, set the nodes of the graph to be all pairs of possible matched cells across the two images. For

Figure 11: Left: original image. Middle: image segmentation and polygon fitting. Right: symmetry cells detected and extracted – an object frame (in RGB color) is attached to each symmetry cell.

each pair of matched pair, we can calculate a camera motion between the two views. Then correct matching should generate the same (up to scale) camera motion. We draw an edge between two nodes if these two pairs of matched cells generate similar camera motions. Therefore, the problem of finding the correctly matched cells becomes the problem of finding the (maximum) cliques in the matching graph. Figure 12 shows the matching results for two cells in three images. The cells that have no match in other images are discarded. The scale for each pair of matched cells are unified.

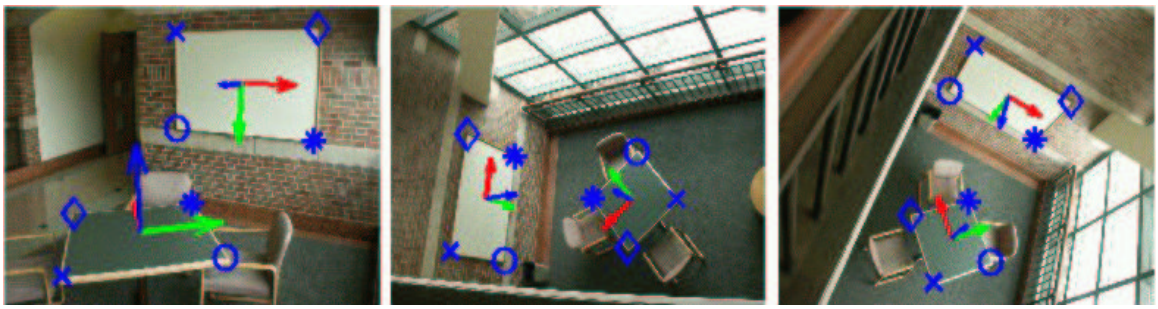

Figure 12: Two symmetry cells are matched in three images. From the raw images, symmetry cell extraction, to cell matching, the process needs no manual intervention.

Reconstruction. Given the set of correctly matched cells, only one more step is needed for reconstruction. Note that the camera motion from different pairs of matched cells have different scales.

To unify the scale, we pick one pair of matched cells as the reference to calculate the translation and use this translation to scale the rest matched pairs. The new scales can further be passed to re-size the cells. Figure 13 shows the reconstructed 3-D cells and the camera poses for the three views. The physical dimension of the cells are calculated with good accuracy. The aspect ratios for the white board and the table top are calculated as 1.51 and 1.01 respectively, with the ground truth being 1.50 and 1.00.

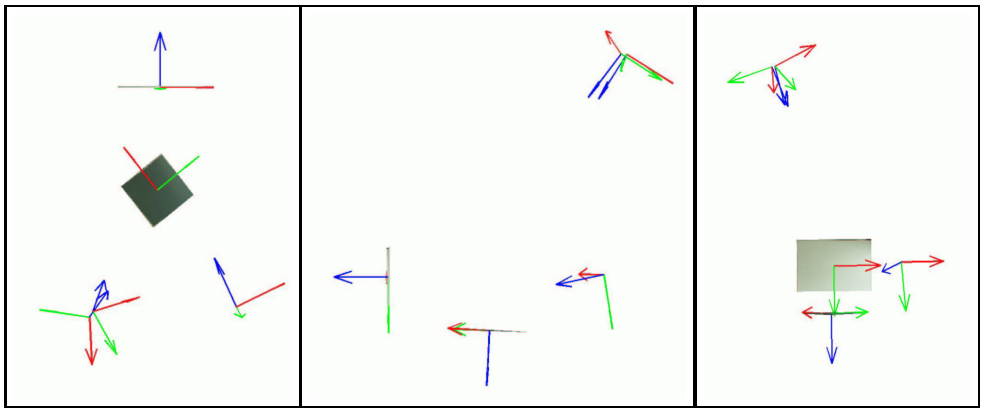

Figure 13: Camera poses and cell structure recovered. From left to right: top, side, and frontal views of the cells and camera poses.

In this example, all the features are matched correctly without any manual intervention. Please note that establishing feature matching among the three views would otherwise be very difficult using other approaches due to the large baseline between the first and second images, and the pure rotational motion between the second and third images. For applications such as robotic mapping, the symmetric cells can serve as "landmarks." The pose and motion of the robot can be easily derived using a similar scheme.

5.3 Semi-automatic building mapping and reconstruction

If a large number of similar objects are present, it is usually hard for the detection and matching scheme in the above example to work properly. For instance, for the symmetry of window complexes on the side of a building shown in Figure 14, many ambiguous matches may occur. In such cases, we need to take manual intervention to obtain a realistic 3-D reconstruction (e.g., see [28]). The techniques discussed so far however help to minimize the amount of manual intervention. For

Figure 14: Five images used for reconstruction of a building. For the first four image, we mark a few cells manually. The last image is only used for extracting roof information.

images in Figure 14, the user only needs to point out cells and provide the cell correspondence information. The system will then automatically generate a consistent set of camera poses from the matched cells, as displayed in Figure 15 top. For each cell, a camera pose is recovered using the symmetry-based algorithm. Similar to the previous example, motion between cameras can be recovered using the corresponding cell. When the camera poses are recovered, the 3-D structure of the building can be recovered as shown in Figure 15 bottom. The 3-D model is rendered as piecewise planar parts. The angles between the normal vectors of any two orthogonal walls differ

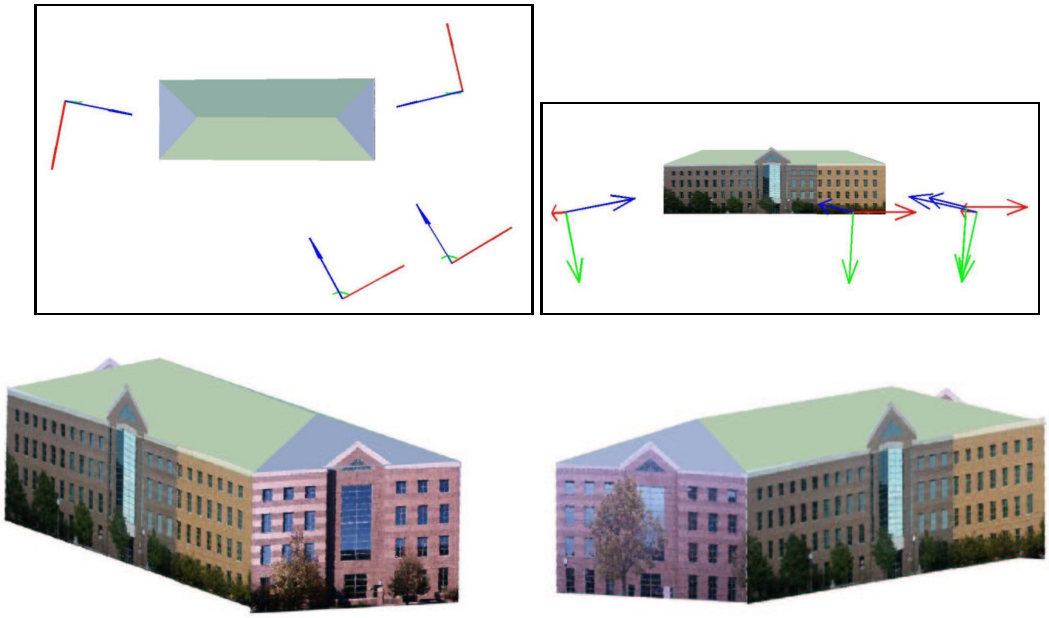

Figure 15: Top: The recovered camera poses and viewpoints as well as the cells. The four coordinate frames are the recovered camera poses from the four images in Figure 14. The roof was substituted by a "virtual" one based on corners extracted from the fifth image with symmetry-based algorithm. Blue arrows are the camera optical axes. Bottom: 3-D model of the building reconstructed from the original set of four images.

from $90°$ by an average of $1°$ error without any nonlinear optimization. The whole process (from taking the images to 3-D rendering) takes less than 20 minutes.

5.4 Summary

The above examples demonstrate the potential of the theory and algorithms introduced in this chapter. Besides these examples, we can expect that these techniques will be very useful in many other applications such as surveillance, manipulation, navigation, and vision-based control. Finally, we like to point out that we are merely at the beginning stage of understanding the geometry

and dynamics associated with visual perception. Much of the topological, geometric, metric, and dynamical relations between 3-D space and 2-D images is still largely unknown, which, to some extent, explains why the capability of existing machine vision systems are still far inferior to that of human vision.

References

[1] A.R.J.Francois, G.G.Medioni, and R.Waupotitsch. Reconstructing mirror symmetric scenes from a single view using 2-view stereo geometry. In *Proceedings of International Conference on Pattern Recognition*, 2002.

[2] S. Avidan and A. Shashua. Novel view synthesis by cascading trilinear tensors. *IEEE Transactions on Visualization and Computer Graphics*, 4(4), 1998.

[3] A.Y.Yang, S.Rao, K. Huang, W. Hong, and Y. Ma. Geometric segmentation of perspective images based on symmetry groups. In *Proceedings of IEEE International Conference on Computer Vision*, Nice, France, 2003.

[4] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 8(6):679–698, 1986.

[5] B. Caprile and V. Torre. Using vanishing points for camera calibration. *International Journal on Computer Vision*, 4(2):127–140, 1990.

[6] S. Carlsson. Symmetry in perspective. In *Proceedings of European Conference on Computer Vision*, pages 249–263, 1998.

[7] P. I. Corke. *Visual Control of Robots: High-Performance Visual Servoing*. Robotics and Mechatronics Series. Research Studies Press LTD, 1996.

[8] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *Proceedings of IEEE International Conference on Computer Vision*, pages 1071–1076, 1995.

[9] A.K. Das, R. Fierro, V. Kumar, B. Southball, J.Spletzer, and C.J.Taylor. Real-time vision-based control of a nonholonomic mobile robot. In *Proceedings of IEEE International Conference and Robotics and Automation*, "2002".

[10] O. Faugeras. *There-Dimensional Computer Vision*. The MIT Press, 1993.

[11] O. Faugeras. Stratification of three-dimensional vision: projective, affine, and metric representations. *Journal of the Optical Society of America*, 12(3):465–84, 1995.

[12] O. Faugeras and Q.-T. Luong. *Geometry of Multiple Images*. The MIT Press, 2001.

[13] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Communications of ACM*, 24(6):381–395, 1981.

[14] J. Gärding. Shape from texture for smooth curved surfaces in perspective projection. *Journal of Mathematical Imaging and Vision*, 2(4):327–350, 1992.

[15] J. Gärding. Shape from texture and contour by weak isotropy. *Journal of Artificial Intelligence*, 64(2):243–297, 1993.

[16] C. Geyer and K. Daniilidis. Properties of the catadioptric fundamental matrix. In *Proceedings of European Conference on Computer Vision*, Copenhagen, Denmark, 2002.

[17] J. Gibson. *The Perception of the Visual World*. Houghton Mifflin, 1950.

[18] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

[19] L. Van Gool, T. Moons, and M. Proesmans. Mirror and point symmetry under perspective skewing. In *International Conference on Computer Vision & Pattern Recognition*, San Francisco, USA, 1996.

[20] A. Gruen and T. Huang. *Calibration and Orientation of Cameras in Computer Vision*. Information Sciences. Springer-Verlag, 2001.

[21] M. Han and T. Kanade. Reconstruction of a scene with multiple linearly moving objects. In *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 542–549, 2000.

[22] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the Alvey Conference*, pages 189–192, 1988.

[23] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge, 2000.

[24] A. Heyden and G. Sparr. Reconstruction from calibrated cameras – a new proof of the Kruppa Demazure theorem. *Journal of Mathematical Imaging and Vision*, pages 1–20, 1999.

[25] W. Hong. Geometry and reconstruction from spatial symmetry. *Master Thesis, UIUC*, July 2003.

[26] B. Horn. *Robot Vision*. MIT Press, 1986.

[27] K. Huang, R. Fossum, and Y. Ma. Generalized rank conditions in multiple view geometry with application to dynamic. In *Proceedings of the 6th European Conference on Computer Vision, Copenhagen, Denmark*, 2002.

[28] K. Huang, W. Hong, Y. Yang, and Y. Ma. Symmetry-based structure from perspecitve images, part ii: Matching and correspondence. *Submitted to IEEE Transactions on Pattern Analysis and Machine Intellegence*, 2003.

[29] T. Huang and O. Faugeras. Some properties of the E matrix in two-view motion estimation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 11(12):1310–12, 1989.

[30] S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, pages 651–670, 1996.

[31] K. Kanatani. *Geometric Computation for Machine Vision*. Oxford Science Publications, 1993.

[32] E. Kruppa. Zur ermittlung eines objecktes aus zwei perspektiven mit innerer orientierung. *Sitz.-Ber.Akad.Wiss., Math.Naturw., Kl.Abt.IIa, 122:1939-1948*, 1913.

[33] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[34] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[35] Y. Ma, K. Huang, and J. Kosecka. New rank deficiency condition for multiple view geometry of line features. *technical report*, May 8, 2001.

[36] Y. Ma, K. Huang, and R. Vidal. Rank deficiency of the multiple view matrix for planar features. *UIUC, CSL Technical Report, UILU-ENG 01-2209 (DC-201)*, May 18, 2001.

[37] Y. Ma, K. Huang, R. Vidal, J. Kosecka, and S. Sastry. Rank conditions of multiple view matrix in multiple view geometry. *International Journal of Computer Vision*, To appear.

[38] Y. Ma, J. Kosecka, and K. Huang. Rank deficiency condition of the multiple view matrix for mixed point and line features. In *Proceedings of Asian Conference on Computer Vision*, Sydney, Australia, 2002.

[39] Y. Ma, J. Kosecka, and S. Sastry. Motion recovery from image sequences: Discrete viewpoint vs. differential viewpoint. In *Proceedings of European Conference on Computer Vision, Volume II*, pages 337–53, 1998.

[40] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3D Vision: From Images to Geometric Models*. Springer-Verlag, 2003.

[41] Y. Ma, Rene Vidal, J. Kosecká, and S. Sastry. Kruppa's equations revisited: its degeneracy, renormalization and relations to cherality. In *Proceedings of European Conference on Computer Vision*, Dublin, Ireland, 2000.

[42] J. Malik and R. Rosenholtz. Computing local surface orientation and shape from texture for curved surfaces. *International Journal on Computer Vision*, 23:149–168, 1997.

[43] D. Marr. *Vision: a computational investigation into the human representation and processing of visual information*. W.H. Freeman and Company, 1982.

[44] S. Maybank. *Theory of Reconstruction from Image Motion*. Springer Series in Information Sciences. Springer-Verlag, 1993.

[45] D. Morales and H. Pashler. No role for colour in symmetry perception. *Nature*, 399:115–116, May 1999.

[46] D. Nister. An efficient solution to the five-point relative pose problem. In *CVPR*, Madison, USA, 2003.

[47] S. E. Plamer. *Vision Science: Photons to Phenomenology*. The MIT Press, 1999.

[48] C. J. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):206–18, 1997.

[49] L. Quan and T. Kanade. A factorization method for affine structure from line correspondences. In *International Conference on Computer Vision & Pattern Recognition*, pages 803–808, 1996.

[50] L. Robert, C. Zeller, O. Faugeras, and M. Hebert. Applications of nonmetric vision to some visually guided tasks. In I. Aloimonos, editor, *Visual Navigation*, pages 89–135, 1996.

[51] O. Shakernia, R. Vidal, C. Sharp, Y. Ma, and S. Sastry. Multiple view motion estimation and control for landing an unmanned aerial vehicle. In *Proceedings of International Conference on Robotics and Automation*, 2002.

[52] A. Shashua. Trilinearity in visual recognition by alignment. In *Proceedings of European Conference on Computer Vision*, pages 479–484. Springer-Verlag, 1994.

[53] A. Shashua and L. Wolf. On the structure and properties of the quadrifocal tensor. In *the Proceedings of ECCV, Volume I*, pages 711–724. Springer-Verlag, 2000.

[54] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[55] I. Shimshoni, Y. Moses, and M. Lindenbaum. Shape reconstruction of 3D bilaterally symmetric surfaces. *International Journal on Computer Vision*, 39:97–112, 2000.

[56] M. Spetsakis and Y. Aloimonos. Structure from motion using line correspondences. *International Journal of Computer Vision*, 4(3):171–184, 1990.

[57] P. Sturm and B. Triggs. A factorizaton based algorithm for multi-image projective structure and motion. In *Proceedings of European Conference on Computer Vision*, pages 709–720. IEEE Comput. Soc. Press, 1996.

[58] S. Thrun. Robotic mapping: A survey. *CMU-CS-02-111*, February 2002.

[59] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography. *Intl. Journal of Computer Vision*, 9(2):137–154, 1992.

[60] B. Triggs. Autocalibration from planar scenes. In *Proceedings of IEEE conference on Computer Vision and Pattern Recognition*, 1998.

[61] N.F. Troje and H.H. Bulthoff. How is bilateral symmetry of human faces used for recognition of novel views. *Vision Research*, 38(1):79–89, 1998.

[62] T. Vetter and T. Poggio. Symmetric 3d objects are an easy case for 2d object recognition. *Spatial Vision*, 8:443–453, 1994.

[63] T. Vetter, T. Poggio, and H. H. Bulthoff. The importance of symmetry and virtual views in three-dimensional object recognition. *Current Biology*, 4:18–23, 1994.

[64] R. Vidal, Y. Ma, S. Hsu, and S. Sastry. Optimal motion estimation from multiview normalized epipolar constraint. In *Proceedings of IEEE International Conference on Computer Vision*, Vancouver, Canada, 2001.

[65] R. Vidal and J. Oliensis. Structure from planar motion with small baselines. In *Proceedings of European Conference on Computer Vision*, pages 383–398, Copenhagen, Denmark, 2002.

[66] R. Vidal, S. Soatto, Y. Ma, and S. Sastry. Segmentation of dynamic scenes from the multibody fundamental matrix. In *Proceedings of ECCV workshop on Vision and Modeling of Dynamic Scenes*, 2002.

[67] J. Weng, T. S. Huang, and N. Ahuja. *Motion and Structure from Image Sequences*. Springer Verlag, 1993.

[68] H. Weyl. *Symmetry*. Princeton Univ. Press, 1952.

[69] A. P. Witkin. Recovering surface shape and orientation from texture. *Journal of Artificial Intelligence*, 17:17–45, 1988.

[70] A.Y. Yang, W. Hong, and Y. Ma. Structure and pose from single images of symmetric objects with applications in robot navigationi. In *Proceedings of International Conference on Robotics and Automation*, Taipei, 2003.

[71] Z. Zhang. A flexible new technique for camera calibration. *Microsoft Technical Report MSR-TR-98-71*, 1998.

[72] W. Y. Zhao and R. Chellappa. Symmetric shape-from-shading using self-ratio image. *International Journal on Computer Vision*, 45(1):55–75, 2001.

[73] A. Zisserman, D. P. Mukherjee, and J. M. Brady. Shape from symmetry-detecting and exploiting symmetry in affine images. *Phil. Trans. Royal Soc. London A*, 351:77–106, 1995.

[74] A. Zisserman, C. A. Rothwell, D. A. Forsyth, and J. L. Mundy. Extracting projective structure from single perspective views of 3d point sets. In *Proceedings of IEEE International Conference on Computer Vision*, 1993.