ΟΠΤΙΚΟΣ ΠΡΟΣΑΝΑΤΟΛΙΣΜΟΣ ΡΟΜΠΟΤΙΚΗΣ ΠΛΑΤΦΟΡΜΑΣ

Η ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

υποβάλλεται στην

ορισθείσα από τη Γενική Συνέλευση Ειδικής Σύνθεσης

Εξεταστική Επιτροπή

του Τμήματος Μηχανικών Η/Υ & Πληροφορικής

από τον

Αλέξανδρο Λιουλεμέ

ως μέρος των Υποχρεώσεων για τη λήψη του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ

ΣΤΙΣ ΤΕΧΝΟΛΟΓΙΕΣ - ΕΦΑΡΜΟΓΕΣ

Πανεπιστήμιο Ιωαννίνων

Ιούλιος 2013

Dedication

I dedicate my dissertation to my parents, Eftalia and Giorgos, who taught me to do what I love, my sister Barbara, who used to argue with me whenever I did something I didn't love, and Antzela, who loves my passion for what I do.

Acknowledgments

I would like to thank my advisor Christophoros Nikou, Assistant Professor of the Department of Computer Science, University of Ioannina, for our very good cooperation during my thesis implementation. Of great importance was also the assistance of Vasilis Karavasilis, PhD student of the Department of Computer Science, University of Ioannina, for his time to upgrade the robotic platform P3-DX. Also, I want to thank the three-member committee, Professor Aristidis Likas and Assistant Professor Konstantinos Blekas for their time to evaluate my thesis. Finally, it will be my omission, to forget all my friends that understand my efforts and my personal ambitions.

Abstract

Lioulemes Alexandros son of Giorgos and Eftalia

MSc, Computer Science Department, University of Ioannina, Greece, July 2013.

Visual homing using a robotic platform.

Thesis Supervisor: Christophoros Nikou.

An algorithm for guiding a robot to its initial position using only a single camera (visual homing) is presented in this thesis. The implemented procedure has as goal to guide a robot to a target position, represented by an image, starting from an arbitrary position which has the initial image in its field of view. Standard visual servoing approaches are based on the epipolar geometry but this model does not consider the planar constraint for points on 2D images. An alternative is the homography-based approach. It behaves well with planar scenes, which are quite usual in man-made environments. At first, scale invariant features (SIFT) are extracted from the target and current images and they are put into correspondence through matching. As the two planar scenes differ by a projective transformation (homography), the first step of the algorithm is to estimate the parameters of the homography. This is accomplished using the four-point algorithm with random sampling consensus (RANSAC) on the set of corresponding points. Then, using epipolar geometry, the homography matrix is decomposed in order to compute the correct rotation matrix and an up-to-scale translation vector which are then used to move the robot. The performance of our visual homing algorithm was validated in an ideal environment and was also tested on data degraded by noise in order to simulate conditions of low resolution cameras and poor illumination.

Περίληψη

Λιουλεμές Αλέξανδρος του Γιώργου και της Ευθαλίας.

MSc, Τμήμα Πληροφορικής Πανεπιστημίου Ιωαννίνων, Ιούλιος 2013.

Οπτικός προσανατολισμός ρομποτικής πλατφόρμας.

Επιβλέποντας: Χριστόφορος Νίκου.

Η παρούσα εργασία πραγματεύεται τον προσανατολισμό ενός ρομπότ χρησιμοποιώντας μόνο μία κοινή κάμερα. Συγκεκριμένα, παρουσιάζεται η υλοποίηση ενός συστήματος που έχει ως σκοπό να οδηγήσει την ρομποτική πλατφόρμα σε μία θέση-στόχο, η οποία αναπαρίσταται από μία 2Δ εικόνα, ξεκινώντας από οποιαδήποτε άλλη θέση, η οποία έχει την αρχική 2Δ εικόνα μέσα στο οπτικό της πεδίο. Επειδή οι δύο εικόνες διαφέρουν κατά έναν προβολικό μετασχηματισμό (ομογραφία), το πρώτο βήμα του αλγόριθμου είναι η εκτίμηση των παραμέτρων της ομογραφίας. Αυτό επιτυγχάνεται με τον αυτόματο εντοπισμό και την αντιστοίχιση χαρακτηριστικών σημείων στις δύο εικόνες. Βασιζόμαστε στους περιγραφείς SIFT οι οποίοι είναι αμετάβλητοι στο χώρο κλίμακας, στην περιστροφή και στον ομοπαραλληλικό μετασηματισμό. Στην συνέχεια, με χρήση της 3Δ επιπολικής γεωμετρίας και της ομογραφίας εκτιμούμε τον πίνακα περιστροφής και το διάνυσμα μετατόπισης μεταξύ των δύο θέσεων της κάμερας του ρομπότ (αρχική και τρέχουσα) και δίνεται στο ρομπότ η εντολή κίνησης με αυτές τις παραμέτρους. Ο αλγόριθμος εφαρμόστηκε επιτυχώς και σε δεδομένα με θόρυβο για την προσομοίωση συνθηκών περιβάλλοντος που αντιμετωπίζονται στην πράξη, όπως είναι η κάμερα χαμηλής ανάλυσης και μελετήθηκε η ακρίβεια των κινήσεων της ρομποτικής πλατφόρμας σε διαφορετικά περιβάλλοντα.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

INTRODUCTION

1.1 Navigation

Robot navigation means the robot's ability to determine its own position in its frame of reference and then to plan a path towards some goal location. In order to navigate in its environment, the robot or any other mobility device requires a representation and the ability to interpret that representation such as to move the robot to the desired position by executing commands. These commands include the required position and orientation, which assume good measurement of the motion obtained by the robot. However, odometry errors or slipping and mechanical drifts may make the desired position not to be reached. Therefore, the use of an additional perception system is mandatory. Vision is perhaps the most broadly researched perception system.

The navigation systems and the determination of motion by images has a large history [8,9] in computer vision and in applications with high precision requirements such as space applications developed by NASA. The usefulness of the orientation system through computer vision is important in applications where an autonomous vehicle is forced to use only its own sensors either of their distance from earth or the inability to use systems like GPS [8–10].

Among different types of sensors, the use of computer vision has some distinct advantages [11]. Unlike orientation sensors (such as GPS and gyros), which provide information only for the movement of the vehicle, the solution of the motion problem of the vehicle through

computer vision methods offers the potential for three-dimensional reconstruction of the environment. Also, in contrast to the use of GPS, which can not work when there is an obstruction between the antenna and the satellite, computer vision can be used for orientation within an urban environment and even indoors. However, computer vision methods several times require high computing power to draw reliable conclusions, while taking multiple images may yield noisy results.

1.2 Related work

There are three different navigation systems based on vision:

- *Map-based navigation*, where some autonomous vehicles are able to execute tasks based on landmarks, which give global localization [12].
- *Map-Building-based navigation*, where the robots carry out specific task, building maps of the enviroment similtaneously [13].
- *Map less navigation*, where robots navigate without landmarks or complex map-building systems [14].

There are different map less navigation systems which are based on panoramic and monocular vision. Most of the homing algorithms implemented by robots capturing panoramic images of their environments. These images are useful because they provide visual information which is not dependent on the orientation of the agent, as would a single field-of-view perspective camera. The robot tracks visual features in panoramic views of the environment that it acquires as it moves [15]. By exploiting only angular information regarding the tracked features, a local control strategy moves the robot between two positions, provided that there are at least three features that can be matched in the panoramas acquired at these positions. The strategy is successful when certain geometric constraints on the configuration of the two positions relative to the features are fulfilled. In order to achieve long-range homing, the features' trajectories are organized in a visual memory during the execution of the "prior" path.

Also, a system for virtual navigation in real environments using image-based panorama rendering. Multiple overlapping images are captured using a camera and a single cube-aligned panorama image is generated for each capture location. Panorama locations are connected in a graph topology and registered with a 2D map for navigation. A real-time image-based viewer renders individual 360-degree panoramas using graphics hardware acceleration. Real-world navigation is performed by traversing the graph and loading new panorama images. The system contains a user-friendly interface and supports standard input and display or a head-mounted display with an inertial tracking device [16].

Other homing algorithms are implemented using monocular vision systems [17]. This system captures images from different positions and uses the discrepancy between this. The target

positions are specified with images taken and memorized in the teaching phase [18]. In the playback phase, the motion correction to reach to the target position is computed from a projective transformation, which is obtained by processing the current image and the stored reference image. The geometric features extracted and matched are lines which have some advantages with respect to points [19], especially in man made environments. This kind of navigation using a representation of the route with a sequence of images has been previously considered by a correlation based matching of the current and the reference images [20]. Extracting and matching geometric information from images is not currently time costly and geometric based approaches are less sensitive to noise or illumination changes than others. Actually, the data of the target images to memorize is lower in method, since only extracted lines are stored. Other authors [21] also use vertical lines to correct robot motion, but using a calibrated trinocular vision system.

Focusing on the topic of visual homing, we present the most important visual algorithm found in the literature. These algorithms fall naturally into two categories: feature-based and image-based (also known as appearance-based). Almost all feature-based algorithms require reliable solutions to the problems of consistent feature extraction and correspondence to ensure successful operation. If these problems can be solved consistently, our review indicates that the epipole-surfing algorithm Basri et al. [22] should be used to home. This algorithm produces an accurate home vector given just two successive "current" images and requires no external compass reference (as several homing algorithms do). Consistent feature extraction, though, is by no means an easy task. It is telling that many experiments in feature-based visual homing take place in adulterated environments with easy-to-detect artifi-cial landmarks. An exception to this was the work of Pons et al. [23]. These authors extracted SIFT [1] features from snapshot and current images and used a robust matching scheme to establish feature correspondence. SIFT features are relatively invariant to changes in orientation, scale and location in images and so are appropriate for the visual homing problem. Though Pons et al. [23] offer impressive homing results in indoor and outdoor static and dynamic environments, their algorithm seems to require thousands of feature similarity computations for each home vector calculation. Pons et al. [23] do not offer figures on how long each home vector computation requires. Computationally efficient SIFT features (e.g. SURFs) [24] have been proposed recently and will probably play a role in future feature-based visual homing algorithms.

Image-based visual homing problems avoid explicit feature correspondence. This approach therefore offers a potentially robust and efficient complement to feature-based homing. Image-based homing schemes infer home vectors by considering every pixel in snapshot and current images, treating each as a feature in its own right. There are three image-based homing algorithms. Image warping requires a computationally intensive brute force search for every home vector computation [25]. We consider this a major drawback. The optical flow-based algorithms [26] though impressive assume that the homing agent is constrained to travel on a single plane. We consider the difference surface homing algorithm pioneered by Zeil et al. [27] to be the image-based homing algorithm which shows the most promise.

The recovery of motion from geometric features has been presented using the epipolar geometry [22]. Rotation and direction of translation are initially computed from the essential matrix. In addition, the steps to collision are also computed using a third image. However, there are situations where the fundamental matrix is not meaningful (small translations or planar scenes) and other models are needed to obtain motion [4] [28].

1.3 Visual Homing

We present a method to guide a mobile robot to locations specified by images previously taken from a home position, which sometimes has been referred as visual homing and can used in conjunction with dead reckoning to allow a robot to explore an area from a given home position and later return to that position, for example, gas-refueling or auto-parking.

Classically this task has been carried out using the fundamental matrix. However the fundamental matrix is ill conditioned with planar scenes, which are quite usual in man made environments. We use a monocular vision system and we compute motion through an homography obtained from point features extracted from the images.

First, we require a homing agent to capture an image at the target position. When seeking to return to target position from a nearby initial position, the agent captures image from initial position and uses the discrepancy between these two to infer the rotation matrix and the translation vector (Figure 1.1).

The overall procedure may be summarized as follows:

- Compute image detectors and descriptors.
- Extract correspondence points between two images.
- Compute the fundamental and planar homography matrix.
- Camera calibration.
- Estimate the essential matrix.
- Decompose the essential and homography matrix.
- Extract camera extrinsic parameters.

Our system consists of five stages, which all must work efficient in order to have a robust visual homing process. These stages are illustrated in Figure 1.1. The first stage extracts features from images with the SIFT algorithm, which transforms image data into scale-invariant coordinates relative to local features, and stores them in a database. The second stage, called correspondence stage, matches each feature from the new image to this previous database and finds candidate matching features based on Euclidean distance of their feature vectors. This

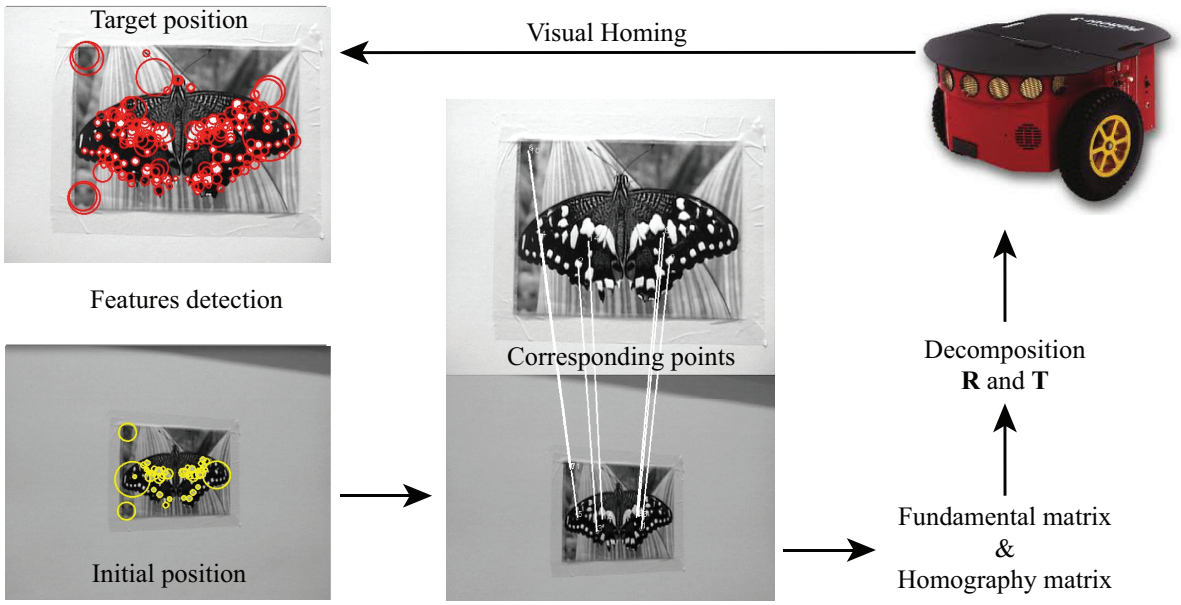


Figure 1.1: Visual Homing in robot navigation.

is performed with the nearest-neighbor algorithm. The third stage estimates the homography matrix from four corresponding points of our two planar scenes, which differ by a projective transformation (homography). In the fourth stage, our system calculate the rotation matrix and the translation vector, using the 3D epipolar geometry and the homography. The fifth stage, implements the movement manipulation of our robotic platform and examines if it is located in the target position. If it is not occurred, our robot feeds back a new current image and makes from the beginning the same visual homing process. Finally, in our thesis we evaluate a low and high visual resolution homing process in order to suggest new perspectives that can be implemented in real indoor environments.

1.4 Structure of the Thesis

In Chapter 2, we present the SIFT algorithm for feature detection and compare it with the SURF. In Chapter 3, we describe the pinhole camera model and solve the problems for finding the intrinsic and extrinsic parameters from that model. In Chapter 4, we introduce the epipolar geometry. First, we explain the meanings of epipole and epipolar lines and estimate the essential matrix. After that, we describe the decomposition process of essential matrix into a rotation matrix and a translation vector and solve the triangulation method in order to recover a four-fold ambiguity. Finally, we estimate the fundamental matrix with the eight-point algorithm and using the RANSAC procedure, we succeed to eliminate pair of outliers. In Chapter 5, we introduce the planar homography and estimate the homography matrix between two planar scenes. Using the homography matrix, we describe the camera calibration method and use it so as to decompose the homography matrix into a fix rotation matrix and a up to scale translation vector. Lastly, in

Chapter 6, we examine the robustness of our visual homing process and evaluate the changes of some parameters.

Chapter 2

FEATURE DETECTION

2.1 Scale Invariant Feature Transform (SIFT)

In this section we will use the SIFT algorithm in order to find the corresponding points in the two images. Image matching is a fundamental aspect of many problems in computer vision, including object or scene recognition, solving for 3D structure from multiple images, stereo correspondence, and motion tracking. The features are invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera viewpoint. They are well localized in both the spatial and frequency domains, reducing the probability of disruption by occlusion, clutter, or noise. Large numbers of features can be extracted from typical images with efficient algorithms. In addition, the features are highly distinctive, which allows a single feature to be correctly matched with high probability against a large database of features, providing a basis for object and scene recognition.

The cost of extracting these features is minimized by taking a cascade filtering approach, in which the more expensive operations are applied only at locations that pass an initial test. Following are the major stages of computation used to generate the set of image features:

- *Scale-space extrema detection*: The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.
- *Keypoint localization*: At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.

- *Orientation assignment*: One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

- *Keypoint descriptor*: The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

This approach has been named the Scale Invariant Feature Transform (SIFT), as it transforms image data into scale-invariant coordinates relative to local features. An important aspect is that it generates large numbers of features that densely cover the image over the full range of scales and locations. A typical image of size 500×500 pixels will give rise to about 2000 stable features (although this number depends on both image content and choices for various parameters). The quantity of features is particularly important for object recognition, where the ability to detect small objects in cluttered backgrounds requires that at least 3 features be correctly matched from each object for reliable identification.

For image matching and recognition, SIFT features are first extracted from a set of reference images and stored in a database. A new image is matched by individually comparing each feature from the new image to this previous database and finding candidate matching features based on Euclidean distance of their feature vectors. We will discuss *fast nearest-neighbor algorithms* that can perform this computation rapidly against large databases.

2.1.1 Detection of Scale-Space Extrema

As described in the introduction, we will detect keypoints using a cascade filtering approach that uses efficient algorithms to identify candidate locations that are then examined in further detail. The first stage of keypoint detection is to identify locations and scales that can be repeatably assigned under differing views of the same object. Detecting locations that are invariant to scale change of the image can be accomplished by searching for stable features across all possible scales, using a continuous function of scale known as scale space.

A good assumption for the scale-space kernel is the Gaussian function. Therefore, the scale space of an image is defined as a function, $L(x, y, \sigma)$, that is produced from the convolution of a variable-scale Gaussian, $G(x, y, \sigma)$, with an input image, $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{2.1}$$

where $*$ is the convolution operation in 2D, and

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{2.2}$$

To efficiently detect stable keypoint locations in scale space, we have proposed using scale-space extrema in the difference-of-Gaussian function convolved with the image, *D(x, y, σ)*, which can be computed from the difference of two nearby scales separated by a constant multiplicative factor k:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned} \tag{2.3}$$

There are a number of reasons for choosing this function. First, it is a particularly efficient function to compute, as the smoothed images, *L*, need to be computed in any case for scale space feature description, and *D* can therefore be computed by simple image subtraction.

In addition, the difference-of-Gaussian function provides a close approximation to the scale-normalized Laplacian of Gaussian, $\sigma^2\nabla^2 G$, showed that the normalization of the Laplacian with the factor σ^2 is required for true scale invariance. In detailed experimental comparisons, the maxima and minima of $\sigma^2\nabla^2 G$ produce the most stable image features compared to a range of other possible image functions, such as the gradient, Hessian, or Harris corner function[reference].

The relationship between *D* and $\sigma^2\nabla^2 G$ can be understood from the heat diffusion equation (parameterized in terms of σ rather than the more usual $t = \sigma^2$):

$$\frac{\partial G}{\partial \sigma} = \sigma\nabla^2 G. \tag{2.4}$$

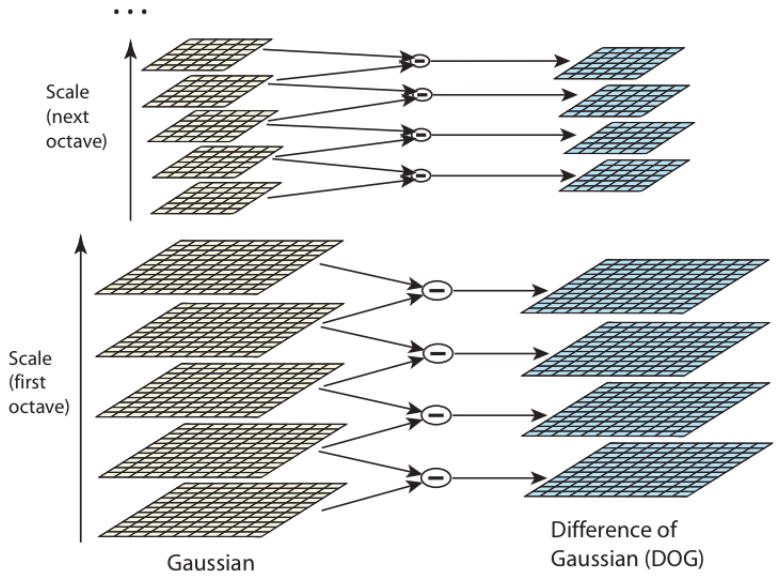


Figure 2.1: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated. The figure was reproduced from [1].

From this, we see that $\nabla^2 G$ can be computed from the finite difference approximation to

$\partial G/\partial\sigma$, using the difference of nearby scales at $k\sigma$ and σ:

$$\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \tag{2.5}$$

and therefore,

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2\nabla^2 G. \tag{2.6}$$

This shows that when the difference-of-Gaussian function has scales differing by a constant factor it already incorporates the σ^2 scale normalization required for the scale-invariant Laplacian. The factor $(k–1)$ in the equation is a constant over all scales and therefore does not influence extrema location. The approximation error will go to zero as k goes to 1.

An efficient approach to construction of $D(x, y, \sigma)$ is shown in Figure 2.1. The initial image is incrementally convolved with Gaussians to produce images separated by a constant factor k in scale space, shown stacked in the left column. We choose to divide each octave of scale space (i.e., doubling of σ) into an integer number, s, of intervals, so $k = 2^{1/s}$. We must produce $s + 3$ images in the stack of blurred images for each octave, so that final extrema detection covers a complete octave. Adjacent image scales are subtracted to produce the difference-of-Gaussian images shown on the right. Once a complete octave has been processed, we resample the Gaussian image that has twice the initial value of σ (it will be 2 images from the top of the stack) by taking every second pixel in each row and column. The accuracy of sampling relative to σ is no different than for the start of the previous octave, while computation is greatly reduced.

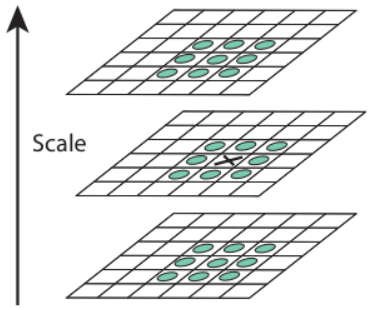


Figure 2.2: Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel (marked with X) to its 26 neighbors in 3×3 regions at the current and adjacent scales (marked with circles). The figure was copied from [1].

2.1.2 Local Extrema Detection

In order to detect the local maxima and minima of $D(x, y, \sigma)$, each sample point is compared to its eight neighbors in the current image and nine neighbors in the scale above and below (see Figure 2.2). It is selected only if it is larger than all of these neighbors or smaller than all of them. The cost of this check is reasonably low due to the fact that most sample points will be eliminated following the first few checks.

An important issue is to determine the frequency of sampling in the image and scale domains that is needed to reliably detect the extrema. Unfortunately, it turns out that there is no minimum

spacing of samples that will detect all extrema, as the extrema can be arbitrarily close together. This can be seen by considering a white circle on a black background, which will have a single scale space maximum where the circular positive central region of the difference-of-Gaussian function matches the size and location of the circle. For a very elongated ellipse, there will be two maxima near each end of the ellipse. As the locations of maxima are a continuous function of the image, for some ellipse with intermediate elongation there will be a transition from a single maximum to two, with the maxima arbitrarily close to each other near the transition.

2.1.3 Accurate Keypoint Localization

Once a keypoint candidate has been found by comparing a pixel to its neighbors, the next step is to perform a detailed fit to the nearby data for location, scale, and ratio of principal curvatures. This information allows points to be rejected that have low contrast (and are therefore sensitive to noise) or are poorly localized along an edge.

A method which has developed [1] to determine the interpolated location of the maximum, uses the Taylor expansion (up to the quadratic terms) of the scale-space function, $D(x, y, \sigma)$, shifted so that the origin is at the sample point:

$$D(x) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}. \tag{2.7}$$

where D and its derivatives are evaluated at the sample point and $\mathbf{x} = (x, y, \sigma)^T$ is the offset from this point. The location of the extremum, $\hat{\mathbf{x}}$, is determined by taking the derivative of this function with respect to $\mathbf{x}$ and setting it to zero, giving

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}. \tag{2.8}$$

The Hessian and derivative of D are approximated by using differences of neighboring sample points. The resulting 3×3 linear system can be solved with minimal cost. If the offset $\hat{\mathbf{x}}$ is larger than 0.5 in any dimension, then it means that the extremum lies closer to a different sample point. In this case, the sample point is changed and the interpolation performed instead about that point. The final offset $\hat{\mathbf{x}}$ is added to the location of its sample point to get the interpolated estimate for the location of the extremum.

The function value at the extremum, $D(\hat{\mathbf{x}})$, is useful for rejecting unstable extrema with low contrast. This can be obtained by substituting Equation 2.8 into 2.7, giving

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}^T \hat{\mathbf{x}}. \tag{2.9}$$

Figure 2.3 shows the effects of keypoint selection on a natural image. In order to avoid too much clutter, a low-resolution 233 by 189 pixel image is used and keypoints are shown as vectors giving the location, scale, and orientation of each keypoint (orientation assignment is

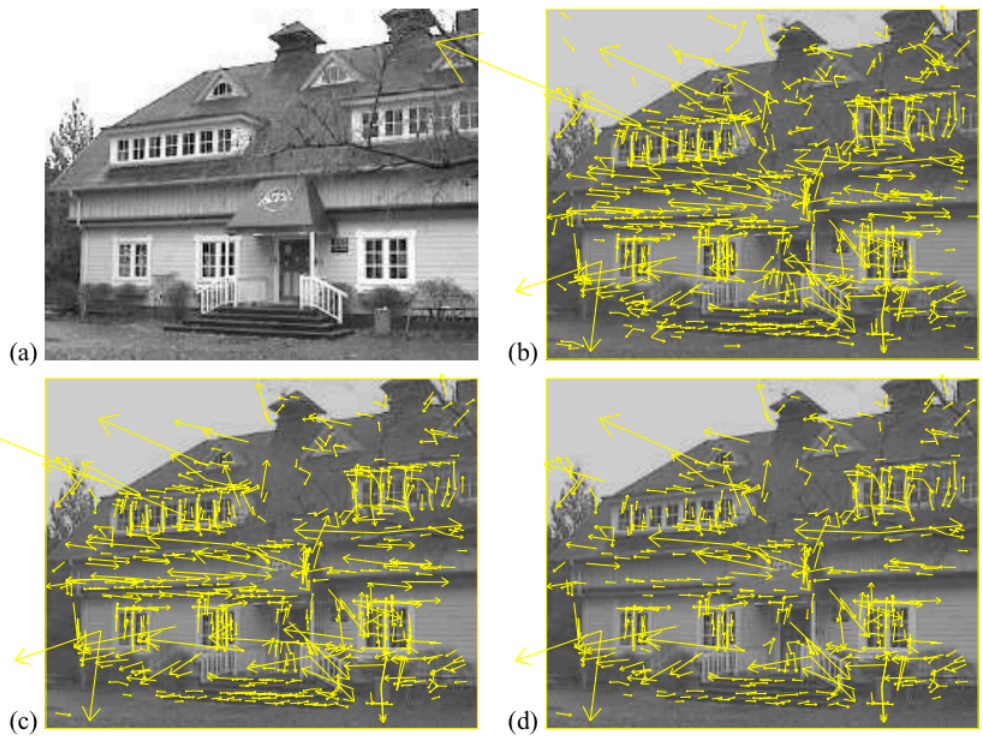


Figure 2.3: This figure shows the stages of keypoint selection. (a) The 233 × 189 pixel original image. (b) The initial 832 keypoints locations at maxima and minima of the difference-of-Gaussian function. Keypoints are displayed as vectors indicating scale, orientation, and location. (c) After applying a threshold on minimum contrast, 729 keypoints remain. (d) The final 536 keypoints that remain following an additional threshold on ratio of principal curvatures. The figure was taken from [1].

described below). Figure 5(a) shows the original image, which is shown at reduced contrast behind the subsequent figures. Figure 5(b) shows the 832 keypoints at all detected maxima and minima of the difference-of-Gaussian function, while (c) shows the 729 keypoints that remain following removal of those with a value of $D(\hat{\mathbf{x}})$ less than 0.03. Part (d) will be explained in the following section.

2.1.4 Eliminating Edge Responses

For stability, it is not sufficient to reject keypoints with low contrast. The difference-of-Gaussian function will have a strong response along edges, even if the location along the edge is poorly determined and therefore unstable to small amounts of noise.

A poorly defined peak in the difference-of-Gaussian function will have a large principal curvature across the edge but a small one in the perpendicular direction. The principal curvatures can be computed from a 2×2 Hessian matrix, $\mathbf{H}$, computed at the location and scale of the keypoint:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.10}$$

The derivatives are estimated by taking differences of neighboring sample points.

The eigenvalues of $\mathbf{H}$ are proportional to the principal curvatures of D. We can avoid explicitly computing the eigenvalues, as we are only concerned with their ratio. Let α be the eigenvalue with the largest magnitude and β be the smaller one. Then, we can compute the sum of the eigenvalues from the trace of $\mathbf{H}$ and their product from the determinant:

$$\begin{aligned} Tr(\mathbf{H}) &= D_{xx} + D_{yy} = \alpha + \beta \\ Det(\mathbf{H}) &= D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta. \end{aligned} \tag{2.11}$$

In the unlikely event that the determinant is negative, the curvatures have different signs so the point is discarded as not being an extremum. Let r be the ratio between the largest magnitude eigenvalue and the smaller one, so that $\alpha = r\beta$. Then

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}, \tag{2.12}$$

which depends only on the ratio of the eigenvalues rather than their individual values. The quantity $(r+1)^2/r$ is at a minimum when the two eigenvalues are equal and it increases with r. Therefore, to check that the ratio of principal curvatures is below some threshold, r, we only need to check

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r+1)^2}{r}. \tag{2.13}$$

2.1.5 Orientation Assignment

By assigning a consistent orientation to each keypoint based on local image properties, the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation. The scale of the keypoint is used to select the Gaussian smoothed image, L, with the closest scale, so that all computations are performed in a scale-invariant manner. For each image sample, $L(x,y)$, at this scale, the gradient magnitude, $m(x,y)$, and orientation, $\theta(x,y)$, is precomputed using pixel differences:

$$\begin{aligned} m(x,y) &= \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \\ \theta(x,y) &= tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \end{aligned} \tag{2.14}$$

An orientation histogram is formed from the gradient orientations of sample points within a region around the keypoint. The orientation histogram has 36 bins covering the 360 degree range of orientations. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint.

Peaks in the orientation histogram correspond to dominant directions of local gradients. The highest peak in the histogram is detected, and then any other local peak that is within 80% of the highest peak is used to also create a keypoint with that orientation. Therefore, for locations

with multiple peaks of similar magnitude, there will be multiple keypoints created at the same location and scale but different orientations. Only about 15% of points are assigned multiple orientations, but these contribute significantly to the stability of matching. Finally, a parabola is fit to the 3 histogram values closest to each peak to interpolate the peak position for better accuracy.

2.1.6 The Local Image Descriptor

The next step is to compute a descriptor for the local image region that is highly distinctive yet is as invariant as possible to remaining variations, such as change in illumination.

One obvious approach would be to sample the local image intensities around the keypoint at the appropriate scale, and to match these using a normalized correlation measure. However, simple correlation of image patches is highly sensitive to changes that cause misregistration of samples, such as affine or 3D viewpoint change or non-rigid deformations.

2.1.7 Descriptor Representation

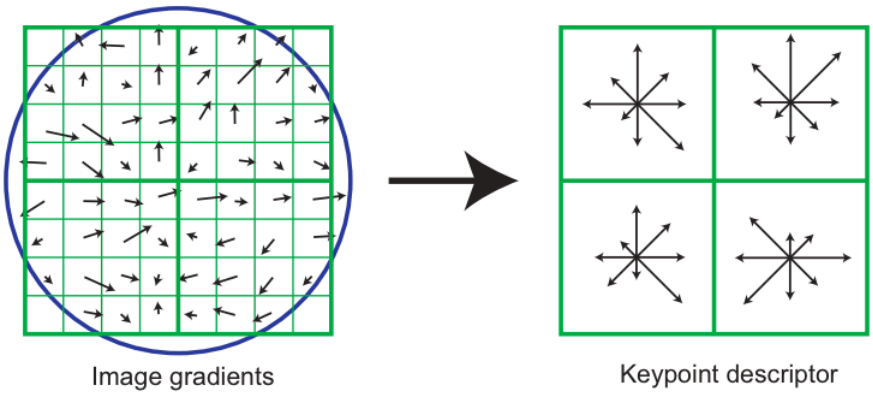


Figure 2.4: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4×4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2×2 descriptor array computed from an 8×8 set of samples, whereas the experiments in this paper use 4×4 descriptors computed from a 16×16 sample array. The figure was reproduced from [1].

Figure 2.4 illustrates the computation of the keypoint descriptor. First the image gradient magnitudes and orientations are sampled around the keypoint location, using the scale of the keypoint to select the level of Gaussian blur for the image. In order to achieve orientation invariance, the coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation.

A Gaussian weighting function with σ equal to one half the width of the descriptor window is used to assign a weight to the magnitude of each sample point. This is illustrated with a circular window on the left side of Figure 2.4, although, of course, the weight falls off smoothly. The purpose of this Gaussian window is to avoid sudden changes in the descriptor with small changes in the position of the window, and to give less emphasis to gradients that are far from the center of the descriptor, as these are most affected by misregistration errors.

The descriptor is formed from a vector containing the values of all the orientation histogram entries, corresponding to the lengths of the arrows on the right side of Figure 2.4. The figure shows a 2×2 array of orientation histograms, whereas our experiments below show that the best results are achieved with a 4×4 array of histograms with 8 orientation bins in each. Therefore, the experiments in this paper use a $4 \times 4 \times 8 = 128$ element feature vector for each keypoint.

2.2 Speeded Up Robust Features (SURF)

In recent years, several scale-invariant features have been proposed and this section presents one of them, the SURF features. SURF stands for *Speeded Up Robust Features*, and as we will see, they are not only scale-invariant features, but they also offer the advantages of being computed very efficiently [24].

In the previous sections, we learned that the image derivatives of an image can be estimated using Gaussian filters. Those filters make use of a σ parameter defining the aperture (size) of the kernel. As we saw, this σ corresponds to the variance of the Gaussian function used to construct the filter, and it then implicitly defines a scale at which the derivative is evaluated. Indeed, a filter having a larger σ value smoothed out the finer details of the image. This is why we can say that it operated at a coarser scale.

Now, if we compute, for instance, the Laplacian of a given image point using Gaussian filters at different scales, then different values are obtained. Looking at the evolution of the filter response for different scale factors, we obtain a curve which eventually reaches a maximum value at some σ value. If we extract this maximum value for two images of the same object taken at two different scales, the ratio of these two σ maxima will correspond to the ratio of the scales at which the images were taken. This important observation is at the core of the scale-invariant feature extraction process. That is, scale-invariant features should be detected as local maxima in both the spatial space (in the image) and the scale space (as obtained from the derivative filters applied at the different scales).

SURF implements this idea by proceeding as follows. First, to detect the features, the Hessian matrix is computed at each pixel (equation 2.10).

The determinant of this matrix gives the strength of this curvature. The idea is therefore to define corners as image points with high local curvature (that is, high variation in more than one direction). Since it is composed of second-order derivatives, this matrix can be computed using Laplacian Gaussian kernels of different scale σ. This Hessian then becomes a function of three

variables: $\mathbf{H}(x, y, \sigma)$. A scale-invariant feature is therefore declared when the determinant of this Hessian reaches a local maximum in both spatial and scale space (that is, 3x3x3 non-maxima suppression needs to be performed).

The calculation of all of these derivatives at different scales is computationally costly. The objective of the SURF algorithm is to make this process as efficient as possible. This is achieved by using approximated Gaussian kernels involving only few integer additions. These have the following structure:

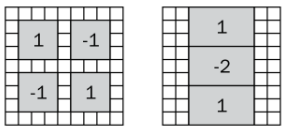


Figure 2.5: Gaussian kernels. The figure was copied from [2].

In figure 2.5 the kernel on the left is used to estimate the mixed second derivatives, while the right one estimates the second derivative in the vertical direction. A rotated version of this second kernel estimates the second derivative in the horizontal direction. The smallest kernels have a size of 9×9 pixels corresponding to $\sigma \approx 1.2$. Kernels of increasing size are successively applied. The exact amount of the filter that is applied can be specified by the additional parameters of the SURF class. By default, 12 different sizes of kernels are used (going up to size 99×99). Note that the fact that integral images are used guarantees that the sum inside each lob can be computed by the using only 3 additions independently of the size of the filter.

Once the local maxima is identified, the precise position of each detected interest point is obtained through interpolation in both scale and image space. The result is then a set of feature points localized at sub-pixel accuracy and to which is associated a scale value.

2.2.1 Describing SURF features

The SURF algorithm, discussed in the preceding recipe, defines a location and a scale for each of the detected features. This scale factor can be used to define the size of a window around the feature point such that the defined neighborhood would include the same visual information no matter what scale the object to which the feature belongs has been pictured. In addition, the visual information included in this neighborhood can be used to characterize the feature point to make it distinguishable from the others.

This recipe will show you how to describe a feature point's neighborhood using compact descriptors. In feature matching, *feature descriptors* are usually N-dimensional vectors that describe a feature point, ideally in a way that is invariant to change in lighting and to small perspective deformations. In addition, good descriptors can be compared using a simple distance metric (for example, Euclidean distance). Therefore, they constitute a powerful tool to use in feature matching algorithms.

Good feature descriptors must be invariant to small changes in illumination, in viewpoint, and to the presence of image noise. Therefore, they are often based on local intensity differences. This is the case of the SURF descriptors which apply the following simple kernels inside a larger neighborhood around a keypoint:

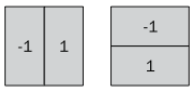


Figure 2.6: Haar wavelet filters. The figure was reproduced from [2].

In figure 2.6 the first one simply measures the local intensity difference in the horizontal direction (designated as *dx*), and the second measures this difference in the vertical direction (designated as *dy*. The size of the neighborhood used to extract the descriptor vector is defined as 20 times the scale factor of the feature (that is, 20σ). This square region is then split into 4×4 smaller square sub-regions. For each sub-region, the kernel responses dx and dy are computed at 5×5 regularly spaced locations (the kernel size being 2σ). All of these responses are summed as follows in order to extract four descriptor values for each subregion:

$$\left[\sum dx \quad \sum dy \quad \sum |dx| \quad \sum |dy|\right] \tag{2.15}$$

Since there are $4 \times 4 = 16$ sub-regions, we have a total of 64 descriptor values. Note that in order to give more importance to the neighboring pixel values closer to the keypoint, the kernel responses are weighted by a Gaussian centered at the keypoint location (with a $\sigma = 3.3$).

The dx and dy responses are also used to estimate the orientation of the feature. These values are computed (with a kernel size of 4σ) within a circular neighborhood of radius 6σ at locations regularly spaced by intervals of σ. For a given orientation, the responses inside a certain angular interval ($\pi/3$) are summed, and the orientation giving the longest vector is defined as the dominant orientation [24].

2.2.2 Descriptors matching

In order to find the corresponding points, we proceed to feature matching by finding the two best matching points for each feature. This is accomplished by performing this matching in two directions, that is, for each point in the first image we find the two best matches in the second image, and then we do the same thing for the feature points of the second image, finding their two best matches in the first image.

Given a feature descriptor q and a set of known feature P, the nearest neighbor of q is the point $p_1 \in P$ with smallest Euclidean distance $||q - p_1||$ (see Figure 2.7). The feature q is assumed to belong to the same class as p_1 if the ration $\frac{p_1}{p_2}$ between the two closest neighbors is smaller that a threshold Θ. In [1], $\Theta = 0.8$ was determined to be a good value for SIFT descriptors, eliminating 90% of the false matches while keeping more than 95% of the correct matches.

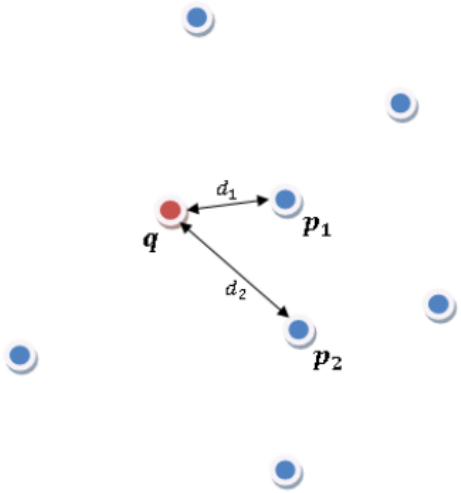


Figure 2.7: A point q and its two nearest neighbors p_1 and p_2, with distances d_1 and d_2 respectively.

The simplest way to find the nearest neighbors is exhaustive or naïve search, where the feature to be classified is directly compared to every element in the database. In figure 2.8 is an example showing the 8 best matches in a match pair containing two images at different view.

2.2.3 SIFT vs SURF

The SIFT algorithm also defined its own descriptor. It is based on the gradient magnitude and orientation computed at the scale of the considered keypoint. As for the SURF descriptors, the scaled neighborhood of the keypoint is divided into 4×4 sub-regions. For each of these regions, an 8-bin histogram of gradient orientations (weighted by their magnitude and by a global Gaussian window centered at the keypoint) is built. Therefore, the descriptor vector is made of the entries of these histograms. There are 4×4 regions and 8 bins per histogram, which leads to a descriptor of length 128.

As for feature detection, the difference between SURF and SIFT descriptors is mainly speed and accuracy. Since SURF descriptors are mostly based on intensity differences, they are faster to compute. However, SIFT descriptors are generally considered to be more accurate in finding the right matching feature.

Figure 2.8: Corresponding points between two different image positions.

Chapter 3

THE PINHOLE CAMERA

This chapter introduces the pinhole or projective camera. This is a purely geometric model that describes the process whereby points in the world are projected into the image. Clearly, the position in the image depends on the position in the world, and the pinhole camera model captures this relationship.

3.1 The pinhole camera

In real life, a pinhole camera consists of a closed chamber with a small hole (the pinhole) in the front (figure 3.1). Rays from an object in the world pass through this hole to form an inverted image on the back face of the box or *image plane*. Our goal is to build a mathematical model of this process.

It is slightly inconvenient that the image from the pinhole camera is upside-down. Hence, we instead consider the *virtual image* that would result from placing the image plane *in front of* the pinhole. Of course, it is not physically possible to build a camera this way, but it is mathematically equivalent to the true pinhole model (except that the image is the right way up) and it is easier to think about. From now on, we will always draw the image plane in front of the pinhole.

Figure 3.2 illustrates the pinhole camera model and defines some terminology. The pinhole itself (the point at which the rays converge) is called the *optical center*. We will assume for

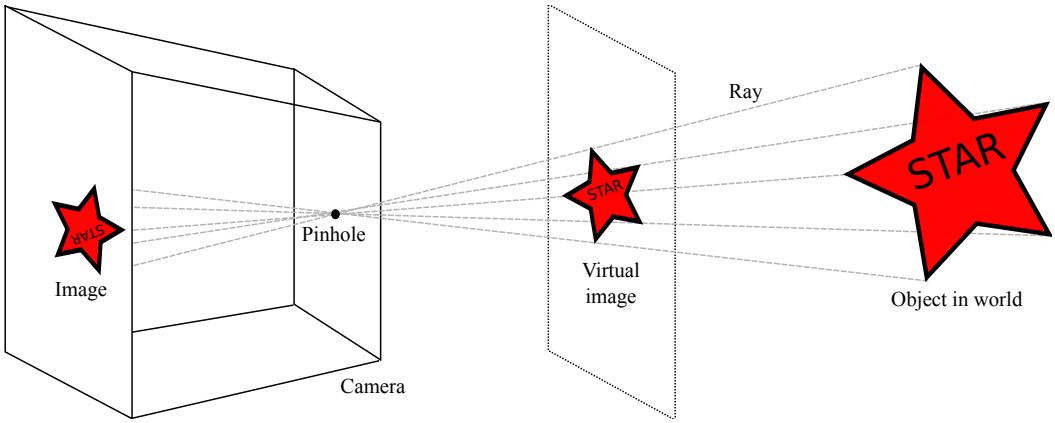


Figure 3.1: The pinhole camera model. Rays from an object in the world pass through the pinhole in the front of the camera and form an image on the back plane (the image plane). This image is upside-down, so we can alternatively consider the virtual image that would have been created if the image plane was in front of the pinhole. This is not physically possible, but it is more convenient to work with. Figure from [3].

now that the optical center is at the origin of the 3D world coordinate system in which points are represented as $\mathbf{w} = [u, v, w]^T$. The virtual image is created on the *image plane*, which is displaced from the optical center along the w-axis or *optical axis*. The point where the optical axis strikes the image plane is known as the *principal point*. The distance between the principal point and the optical center (i.e., the distance between the image plane and the pinhole) is known as the *focal length*.

The pinhole camera model is a generative model that describes the likelihood $Pr(\mathbf{x}|\mathbf{w})$ of observing a feature at position $\mathbf{x} = [x, y]^T$ in the image, given that it is the projection of a 3D point $\mathbf{w} = [u, v, w]^T$ in the world. Although light transport is essentially deterministic, we will nonetheless build a probability model; there is noise in the sensor, and unmodeled factors in the feature detection process can also affect the measured image position. However, for pedagogical reasons we will defer a discussion of this uncertainty until later, and temporarily treat the imaging process as if it were deterministic.

Our task then is to establish the position $\mathbf{x} = [x, y]^T$ where the 3D point $\mathbf{w} = [u, v, w]^T$ is imaged. Considering figure 3.2, it is clear how to do this. We connect a ray between $\mathbf{w}$ and the optical center. The image position $\mathbf{x}$ can be found by observing where this ray strikes the image plane. This process is called *perspective projection*. In the next few sections, we will build a more precise mathematical model of this process. We will start with a very simple camera model (the normalized camera) and build up to a full camera parameterization.

3.1.1 The normalized camera

In the *normalized camera*, the focal length is one, and it is assumed that the origin of the 2D coordinate system (*x*,*y*) on the image plane is centered at the principal point. Figure 3.3 shows a 2D slice of the geometry of this system (the *u*- and *x*-axes now point upward out of the page

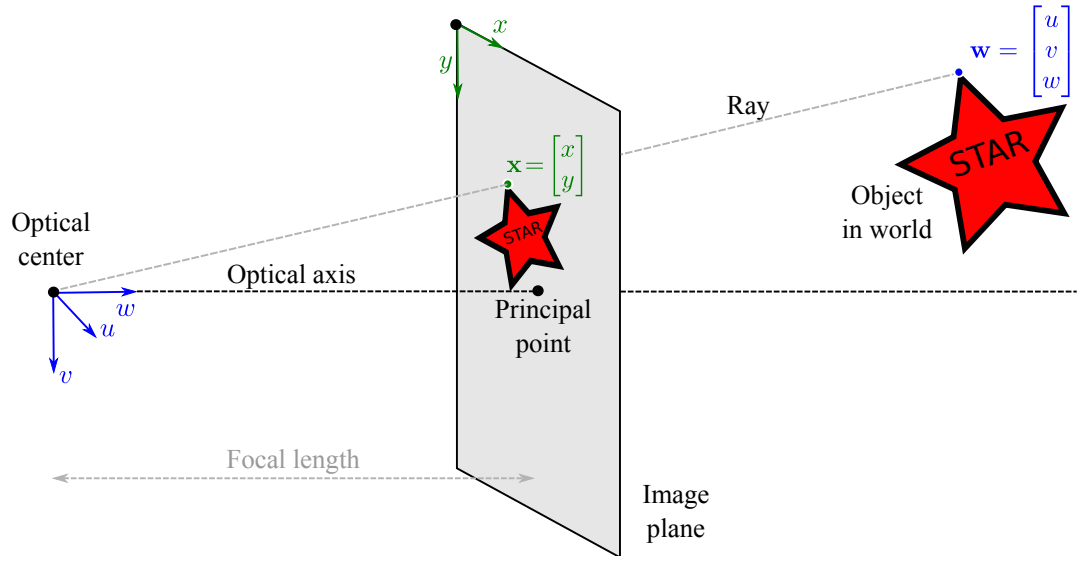


Figure 3.2: Pinhole camera model terminology. The optical center (pinhole) is placed at the origin of the 3D world coordinate system (u, v, w), and the image plane (where the virtual image is formed) is displaced along the w-axis, which is also known as the optical axis. The position where the optical axis strikes the image plane is called the principal point. The distance between the image plane and the optical center is called the focal length. The figure was copied from [3].

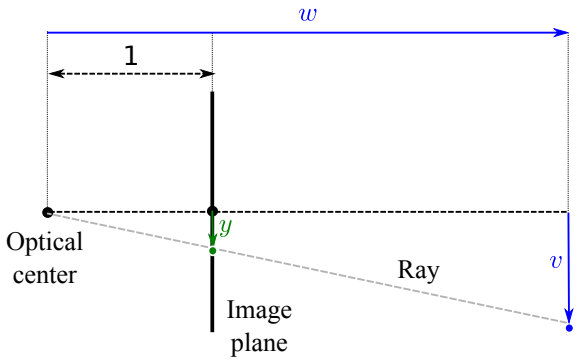


Figure 3.3: Normalized camera. The focal length is one, and the 2D image coordinate system (x,y) is centered on the principal point (only y- axis shown). By similar triangles, the y position in the image of a point at (u, v, w) is given by v/w. This corresponds to our intuition: as an object gets more distant, its projection becomes closer to the center of the image. Figure from [3].

and cannot be seen). By similar triangles, it can easily be seen that the y-position in the image of the world point at $\mathbf{w} = [u, v, w]^T$ is given by v/w. More generally,in the normalized camera, a 3D point $\mathbf{w} = [u, v, w]^T$ is projected into the image at $\mathbf{x} = [x, y]^T$ using the relations

$$x = \frac{u}{w}, \quad y = \frac{v}{w} \tag{3.1}$$

where x, y, u, v and w are measured in the same real-world units (e.g., mm).

3.1.2 Focal length parameters

The normalized camera is unrealistic; for one thing, in a real camera, there is no particular reason why the focal length should be one. Moreover, the final position in the image is measured in pixels, not physical distance, so the model must take into account the photoreceptor spacing. Both of these factors have the effect of changing the mapping between points $\mathbf{w} = [u, v, w]^T$ in

the 3D world and their 2D positions $\mathbf{x} = [x, y]^T$ in the image plane by a constant scaling factor ϕ (figure 3.4), so that

$$x = \frac{\phi u}{w}, \quad y = \frac{\phi v}{w}. \tag{3.2}$$

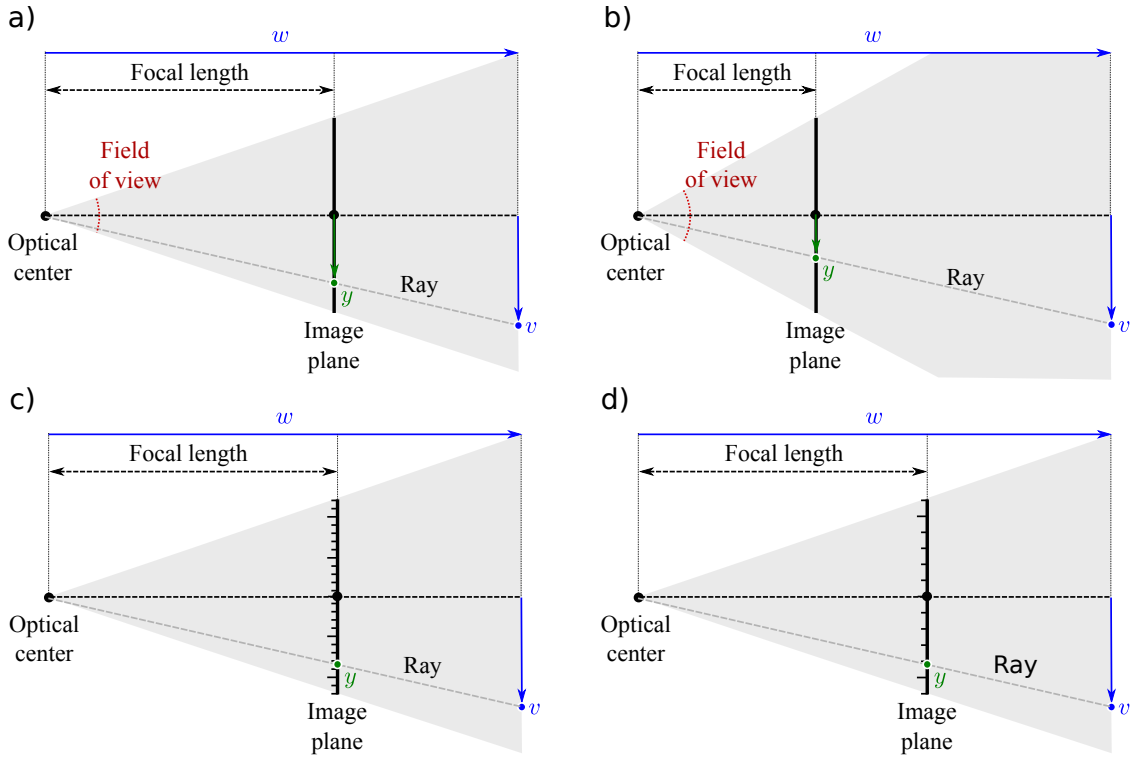


Figure 3.4: Focal length and photoreceptor spacing. a-b) Changing the distance between the optical center and the image plane (the focal length) changes the relationship between the 3D world point **w** and the 2D image point **x**. In particular, if we take the original focal length (a) and halve it (b), the 2D image coordinate is also halved. The *field of view* of the camera is the total angular range that is imaged (usually different in the x- and y-directions). When the focal length decreases, the field of view increases. c-d) The position in the image **x** is usually measured in pixels. Hence, the position **x** depends on the density of the receptors on the image plane. If we take the original photoreceptor density (c) and halve it (d), then the 2D image coordinate is also halved. Hence, the photoreceptor spacing and focal length both change the mapping from rays to pixels in the same way. The figure was reproduced from [3].

To add a further complication, the spacing of the photoreceptors may differ in the x- and y-directions, and so the scaling may be different in each direction, giving the relations

$$x = \frac{\phi_x u}{w}, \quad y = \frac{\phi_y v}{w}. \tag{3.3}$$

where ϕ_x and ϕ_y are separate scaling factors for the x- and y-directions. These parameters are known as the *focal length parameters* in the x- and y-directions, but this name is somewhat misleading - they account for not just the distance between the optical center and the principal point (the true focal length) but also the photoreceptor spacing.

3.1.3 Offset and skew parameters

The model so far is still incomplete in that pixel position $\mathbf{x} = [0, 0]^T$ is at the principal point (where the w-axis intersects the image plane). In most imaging systems, the pixel position $\mathbf{x} = [0, 0]^T$ is at the top-left of the image rather than the center. To cope with this, we add *offset parameters* δ_x and δ_y so that

$$x = \frac{\phi_x u}{w} + \delta_x, \quad y = \frac{\phi_y v}{w} + \delta_y. \tag{3.4}$$

where δ_x and δ_y are the offsets in pixels from the top-left corner of the image to the position where the *w* axis strikes the image plane. Another way to think about this is that the vector $[\delta_x, \delta_y]^T$ is the position of the principal point in pixels.

If the image plane is exactly centered on the *w*-axis, these offset parameters should be half the image size: for a 640 × 480 VGA image δ_x and δ_y would be 320 and 240, respectively. However, in practice it is difficult and superfluous to manufacture cameras with the imaging sensor perfectly centered, and so we treat the offset parameters as variable quantities.

We also introduce a *skew* term γ which moderates the projected position *x* as a function of the height v in the world. This parameter has no clear physical interpretation, but can help explain the projection of points into the image in practice. The resulting camera model is

$$x = \frac{\phi_x u + \gamma v}{w} + \delta_x, \quad y = \frac{\phi_y v}{w} + \delta_y. \tag{3.5}$$

3.1.4 Position and the orientation of camera

Finally, we must account for the fact that the camera is not always conveniently centered at the origin of the world coordinate system with the optical axis exactly aligned with the *w*-axis. In general, we may want to define an arbitrary world coordinate system that may be common to more than one camera. To this end, we express the world points *w* in the coordinate system of the camera before they are passed through the projection model, using the coordinate transformation:

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \tag{3.6}$$

or

$$\mathbf{w}' = \mathbf{R}\mathbf{w} + \mathbf{T}, \tag{3.7}$$

where $\mathbf{w}'$ is the transformed point, $\mathbf{R}$ is a 3×3 rotation matrix, and $\mathbf{T}$ is a 3×1 translation matrix.

3.1.5 Full pinhole camera model

We are now in a position to describe the full camera model, by combining equations 3.5 and 3.6. A 3D point $\mathbf{w}=[u,v,w]^T$ is projected to a 2D point $\mathbf{x}=[x,y]^T$ by the relations

$$\begin{aligned} x &= \frac{\phi_x(r_{11}u + r_{12}v + r_{13}w + t_x) + \gamma(r_{21}u + r_{22}v + r_{23}w + t_y)}{r_{31}u + r_{32}v + r_{33}w + t_z} + \delta_x \\ y &= \frac{\phi_y(r_{21}u + r_{22}v + r_{23}w + t_y)}{r_{31}u + r_{32}v + r_{33}w + t_z} + \delta_y. \end{aligned} \tag{3.8}$$

There are two sets of parameters in this model. The *intrinsic* or *camera parameters* $\{\phi_x, \phi_y, \gamma, \delta_x, \delta_y\}$ describe the camera itself, and the *extrinsic parameters* $\{\mathbf{R}, \mathbf{T}\}$ describe the position and orientation of the camera in the world. For reasons that will become clear in section 3.3.1, we will store the intrinsic parameters in the *intrinsic matrix* $\mathbf{\Lambda}$ where

$$\mathbf{\Lambda} = \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.9}$$

We can now abbreviate the full projection model (equations 3.8) by just writing

$$\mathbf{x} = \textbf{pinhole}[\mathbf{w}, \mathbf{\Lambda}, \mathbf{R}, \mathbf{T}]. \tag{3.10}$$

Finally, we must account for the fact that the estimated position of a feature in the image may differ from our predictions. There are a number of reasons for this, including noise in the sensor, sampling issues, and the fact that the detected position in the image may change at different viewpoints. We model these factors with additive noise that is normally distributed with a spherical covariance to give the final relation

$$Pr(\mathbf{x}|\mathbf{w}, \mathbf{\Lambda}, \mathbf{R}, \mathbf{T}) = \mathsf{Norm}_{\mathbf{x}}[\textbf{pinhole}[\mathbf{w}, \mathbf{\Lambda}, \mathbf{R}, \mathbf{T}], \sigma^2\mathbf{I}], \tag{3.11}$$

where σ^2 is the variance of the noise.

The *multivariate normal* or Gaussian distribution models D-dimensional variables $\mathbf{x}$ where each of the D elements $x_1 \ldots x_D$ is continuous and lies in the range $[-\infty, +\infty]$. As such the univariate normal distribution is a special case of the multivariate normal where the number of elements D is one. In machine vision the multivariate normal might model the joint distribution of the intensities of D pixels within a region of the image. The state of the world might also be described by this distribution. For example, the multivariate normal might describe the joint uncertainty in the 3D position (x, y, z) of an object in the scene.

The multivariate normal distribution has two parameters: the mean $\boldsymbol{\mu}$ and covariance $\mathbf{\Sigma}$. The mean $\boldsymbol{\mu}$ is a $D \times 1$ vector that describes the mean of the distribution. The covariance $\mathbf{\Sigma}$ is a symmetric $D \times D$ positive definite matrix so that $\mathbf{z}^T\mathbf{\Sigma}\mathbf{z}$ is a positive for any real vector $\mathbf{z}$. The

probability density function has the following form

$$Pr(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}|\mathbf{\Sigma}|^{1/2}}\exp[-0.5(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})], \tag{3.12}$$

or for short

$$Pr(\mathbf{x}) = \text{Norm}_{\mathbf{x}}[\boldsymbol{\mu}, \mathbf{\Sigma}]. \tag{3.13}$$

Note that the pinhole camera is a *generative model*. We are describing the likelihood $Pr(\mathbf{x}|\mathbf{w}, \mathbf{\Lambda}, \mathbf{R}, \mathbf{T})$ of observing a 2D image point $\mathbf{x}$ given a 3D world point $\mathbf{w}$ and the parameters $\{\mathbf{\Lambda}, \mathbf{R}, \mathbf{T}\}$

3.1.6 Radial distortion

In the previous section, we introduced the pinhole camera model. However, it has probably not escaped your attention that real-world cameras are rarely based on the pinhole: they have a lens (or possibly a system of several lenses) that collects light from a larger area and re-focuses it on the image plane. In practice, this leads to a number of deviations from the pinhole model. For example, some parts of the image may be out of focus, which essentially means that the assumption that a point in the world $\mathbf{w}$ maps to a single point in the image $\mathbf{x}$ is no longer valid.

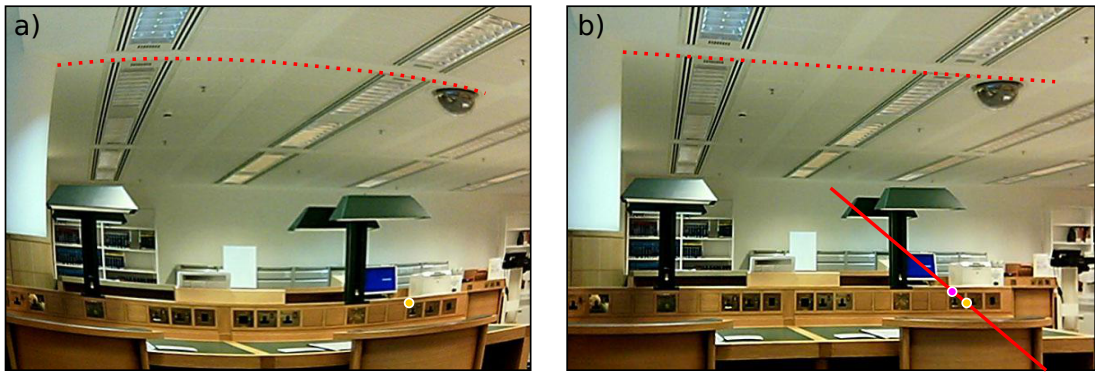


Figure 3.5: Radial distortion. a) An image that suffers from radial distortion is easily spotted because lines that were straight in the world are mapped to curves in the image (e.g., red dotted line). b) After applying the inverse radial distortion model, straight lines in the world now correctly map to straight lines in the image. The distortion caused the magenta point to move along the red radial line to the position of the yellow point. The figure was taken from [3].

However, there is one deviation from the pinhole model that must be addressed. *Radial distortion* is a nonlinear warping of the image that depends on the distance from the center of the image. In practice, this occurs when the field of view of the lens system is large. It can easily be detected in an image, because straight lines in the world no longer project to straight lines in the image (figure 3.5).

Radial distortion is commonly modeled as a polynomial function of the distance r from the center of the image. In the normalized camera, the final image positions (x',y') are expressed as

functions of the original positions (x, y) by

$$\begin{aligned} x' &= x(1 + \beta_1 r^2 + \beta_2 r^4) \\ y' &= y(1 + \beta_1 r^2 + \beta_2 r^4), \end{aligned} \tag{3.14}$$

where the parameters β_1 and β_2 control the degree of distortion. These relations describe a family of possible distortions that approximate the true distortion closely for most common lenses.

This distortion is implemented after perspective projection (division by w) but before the effect of the intrinsic parameters (focal length offset, etc.), so the warping is relative to the optical axis and not the origin of the pixel coordinate system. We will not discuss radial distortion further in this volume.

3.2 Geometric problems

Now that we have described the pinhole camera model, we will consider two important geometric problems. Each is an instance of learning or inference within this model.

3.2.1 Problem 1: Learning extrinsic parameters

We aim to recover the position and orientation of the camera relative to a known scene. This is sometimes known as the *perspective-n-point (PnP)* problem or the *exterior orientation* problem. One common application is augmented reality, where we need to know this relationship to render virtual objects that appear to be stable parts of the real scene.

The problem can be stated more formally as follows: we are given a known object, with I distinct 3D points $\{\mathbf{w}_i\}_{i=1}^I$, their corresponding projections in the image $\{\mathbf{x}_i\}_{i=1}^I$, and known intrinsic parameters $\mathbf{\Lambda}$. Our goal is to estimate the rotation $\mathbf{R}$ and translation $\mathbf{T}$ that map points in the coordinate system of the object to points in the coordinate system of the camera so that

$$\hat{\mathbf{R}}, \hat{\mathbf{T}} = \arg\max_{\mathbf{R},\mathbf{T}} \left[\sum_{i=1}^{I} log[Pr(\mathbf{x}_i|\mathbf{w}_i, \mathbf{\Lambda}, \mathbf{R}, \mathbf{T})] \right]. \tag{3.15}$$

This is a maximum likehood learning problem, in which we aim to find parameters $\mathbf{R}, \mathbf{T}$ that make the predictions $\textbf{pinhole}[\mathbf{w}_i, \mathbf{\Lambda}, \mathbf{R}, \mathbf{T}]$ of the model agree with the observed 2D points $\mathbf{x}_i$.

3.2.2 Problem 2: Learning intrinsic parameters

We aim to estimate the intrinsic parameters $\mathbf{\Lambda}$ hat relate the direction of rays through the optical center to coordinates on the image plane. This estimation process is known as *calibration*. Knowledge of the intrinsic parameters is critical if we want to use the camera to build 3D models of the world.

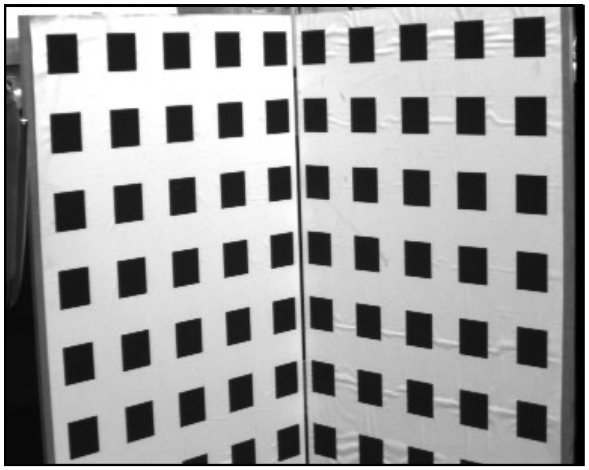


Figure 3.6: Camera calibration target. One way to calibrate the camera (estimate its intrinsic parameters) is to view a 3D object (a camera calibration target) for which the geometry is known. The marks on the surface are at known 3D positions in the frame of reference of the object, and are easy to locate in the image using basic image-processing techniques. It is now possible to find the intrinsic and extrinsic parameters that optimally map the known 3D positions to their 2D projections in the image. On chapter 5 it is noted that calibration is more usually based on a number of views of a known 2D planar object. The figure was taken from [4].

The calibration problem can be stated formally as follows: given a known 3D object, with I distinct 3D points $\{\mathbf{w}_i\}_{i=1}^{I}$ and their corresponding projections in the image $\{\mathbf{x}_i\}_{i=1}^{I}$, estimate the intrinsic parameters:

$$\hat{\mathbf{\Lambda}} = \arg\max_{\mathbf{\Lambda}} \left[\max_{\mathbf{R},\mathbf{T}} \left[\sum_{i=1}^{I} log[Pr(\mathbf{x}_i|\mathbf{w}_i, \mathbf{\Lambda}, \mathbf{R}, \mathbf{T})] \right] \right]. \tag{3.16}$$

Once more, this is a maximum likelihood learning problem in which we aim to find parameters $\mathbf{\Lambda}, \mathbf{R}, \mathbf{T}$ that make the predictions of the model $\textbf{pinhole}[\mathbf{w}_i, \mathbf{\Lambda}, \mathbf{R}, \mathbf{T}]$ agree with the observed 2D points $\mathbf{x}_i$. We do not particularly care about the extrinsic parameters $\mathbf{R}, \mathbf{T}$; finding these is just a means to the end of estimating the intrinsic parameters $\mathbf{\Lambda}$.

The calibration process requires a known 3D object, on which distinct points can be identified, and their projections in the image found. A common approach is to construct a bespoke 3D *calibration target* that achieves these goals (figure 3.6).

3.2.3 Solving the problems

We have introduced two geometric problems, each of which took the form of a learning or inference problem using the pinhole camera model. We formulated each in terms of maximum likelihood estimation, and in each case this results in an optimization problem.

Unfortunately, none of the resulting objective functions can be optimized in closed form; each solution requires the use of nonlinear optimization. In each case, it is critical to have a good initial estimate of the unknown quantities to ensure that the optimization process converges to the global maximum. In the remaining part of this chapter we develop algorithms that provide these initial estimates. The general approach is to choose new objective functions that can be optimized in closed form, and where the solution is close to the solution of the true problem.

3.3 Homogeneous coordinates

To get good initial estimates of the geometric quantities in the preceding optimization problems, we play a simple trick: we change the representation of both the 2D image points and 3D world points so that the projection equations become linear. After this change, it is possible to find solutions for the unknown quantities in closed form. However, it should be emphasized that these solutions do not directly address the original optimization criteria: they minimize more abstract objective functions based on *algebraic error* whose solutions are not guaranteed to be the same as those for the original problem. However, they are generally close enough to provide a good starting point for a nonlinear optimization of the true cost function.

We convert the original Cartesian representation of the 2D image points $\mathbf{x}$ to a 3D *homogeneous* coordinate $\tilde{\mathbf{x}}$ so that

$$\tilde{\mathbf{x}} = \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{3.17}$$

where λ is an arbitrary scaling factor. This is redundant representation in that any scalar multiple λ represents the same 2D point. For example, the homogeneous vectors $\tilde{\mathbf{x}} = [2, 4, 2]^T$ and $\tilde{\mathbf{x}} = [3, 6, 3]^T$ both represent the Cartesian 2D point $\mathbf{x} = [1, 2]^T$, where scaling factors $\lambda = 2$ and $\lambda = 3$ have been used, respectively.

Converting between homogeneous and Cartesian coordinates is easy. To move to homogeneous coordinates, we choose $\lambda = 1$ and simply append a 1 to the original 2D Cartesian coordinate. To recover the Cartesian coordinates, we divide the first two entries of the homogeneous 3-vector by the third, so that if we observe the homogeneous vector $\tilde{\mathbf{x}} = [x, \tilde{y}, z]^T$, then we can recover the Cartesian coordinate $\mathbf{x} = [x, y]^T$ as

$$x = \frac{\tilde{x}}{\tilde{z}}, \quad y = \frac{\tilde{y}}{\tilde{z}}. \tag{3.18}$$

Further insight into the relationship between the two representations is given in figure 3.7.

It is similarly possible to represent the 3D world point $\mathbf{w}$ as a homogeneous 4D vector $\tilde{\mathbf{w}}$ so that

$$\tilde{\mathbf{w}} = \lambda \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}, \tag{3.19}$$

where λ is again an arbitrary scaling factor. Once more, the conversion from Cartesian to homogeneous coordinates can be achieved by appending a 1 to the original 3D vector $\mathbf{w}$. The conversion from homogeneous to Cartesian coordinates is achieved by dividing each of the first three entries by the last.

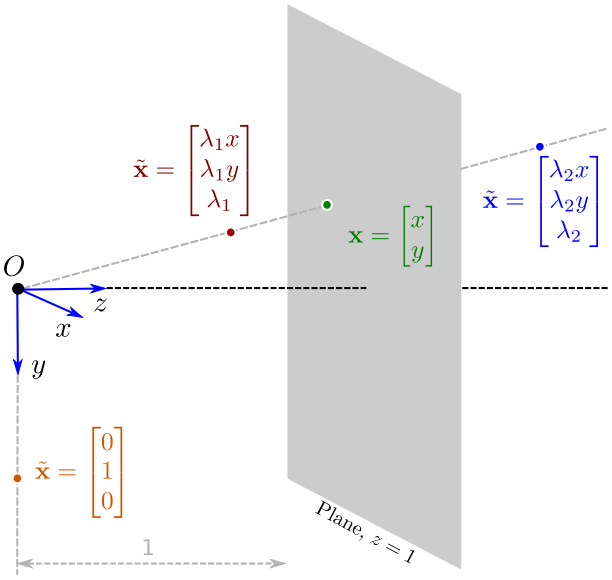


Figure 3.7: Geometric interpretation of homogeneous coordinates. The different scalar multiples λ of the homogeneous 3-vector $\tilde{\mathbf{x}}$ define a ray through the origin of a coordinate space. The corresponding 2D image point $\mathbf{x}$ can be found by considering the 2D point that this ray strikes on the plane at $z = 1$. An interesting side-effect of this representation is that it is possible to represent points at infinity (known as *ideal points*). For example, the homogeneous coordinate defines a ray $[0, 1, 0]^T$ that is parallel to $z = 1$ and so never intersects the plane. It represents the point at infinity in direction $[0, 1]^T$. The figure was reproduced from [3].

3.3.1 Camera model in homogeneous coordinates

It is hard to see the point of converting the 2D image points to homogeneous 3-vectors and converting the 3D world point to homogeneous 4-vectors until we re-examine the pinhole projection equations,

$$\begin{aligned} x &= \frac{\phi_x u + \gamma v}{w} + \delta_x \\ y &= \frac{\phi_y v}{w} + \delta_y, \end{aligned} \tag{3.20}$$

where we have temporarily assumed that the world point $\mathbf{w} = [u, v, w]^T$ same coordinate system as the camera.

In homogeneous coordinates, these relationships can be expressed as a set of linear equations

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x & 0 \\ 0 & \phi_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}. \tag{3.21}$$

To convince ourselves of this, let us write these relations explicitly:

$$\begin{aligned} \lambda x &= \phi_x u + \gamma v + \delta_x w \\ \lambda y &= \phi_y v + \delta_y w \\ \lambda &= w. \end{aligned} \tag{3.22}$$

We solve for x and y by converting back to Cartesian coordinates: we divide the first two relations by the third to yield the original pinhole model (equation 3.20)

Let us summarize what has happened: the original mapping from 3D Cartesian world points to 2D Cartesian image points is nonlinear (due to the division by w). However, the mapping from 4D homogeneous world points to 3D homogeneous image points is linear. In the homogeneous representation, the nonlinear component of the projection process (division by w) has been side-stepped: this operation still occurs, but it is in the final conversion back to 2D Cartesian coordinates, and thus does not trouble the homogeneous camera equations.

To complete the model, we add the extrinsic parameters $\{\mathbf{R}, \mathbf{T}\}$ that relate the world coordinate system and the camera coordinate system, so that

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x & 0 \\ 0 & \phi_y & \delta_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}, \tag{3.23}$$

or in matrix mode

$$\lambda \tilde{\mathrm{x}} = \begin{bmatrix} \mathbf{\Lambda} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}^T & 1 \end{bmatrix} \tilde{\mathbf{w}}, \tag{3.24}$$

where $\mathbf{0} = [0, 0, 0]^T$. The same relations can be simplified to

$$\lambda \tilde{\mathbf{x}} = \mathbf{\Lambda} \begin{bmatrix} \mathbf{R} & \mathbf{T} \end{bmatrix} \tilde{\mathbf{w}}. \tag{3.25}$$

In the next two sections, we revisit the two geometric problems. In each case, we will use algorithms based on homogeneous coordinates to compute good initial estimates of the variable of interest. These estimates can then be improved using nonlinear optimization.

3.4 Learning extrinsic parameters

Given a known object, with *I* distinct 3D points $\{\mathbf{w}_i\}_{i=1}^I$, the corresponding projections in the image $\{\mathbf{x}_i\}_{i=1}^I$, and known intrinsic parameters $\mathbf{\Lambda}$, estimate the geometric relationship between the camera object determined by the rotation $\mathbf{R}$ and the translation $\mathbf{T}$,

$$\hat{\mathbf{R}}, \hat{\mathbf{T}} = \arg\max_{\mathbf{R},\mathbf{T}} \left[\sum_{i=1}^{I} log[Pr(\mathbf{x}_i | \mathbf{w}_i, \mathbf{\Lambda}, \mathbf{R}, \mathbf{T})] \right]. \tag{3.26}$$

This is a non-convex problem, so we make progress by expressing it in homogeneous coordinates. The relationship between the i^{th} homogeneous world point $\tilde{\mathbf{w}}_i$ and the i^{th} corresponding

homogeneous image point $\tilde{\mathbf{x}}_i$ is

$$\lambda_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}, \tag{3.27}$$

We would like to discard the effect of the (known) intrinsic parameters $\mathbf{\Lambda}$. To this end, we pre-multiply both sides of the equation by the inverse of the intrinsic matrix $\mathbf{\Lambda}$ to yield

$$\lambda_i \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}. \tag{3.28}$$

The transformed coordinates $\tilde{\mathbf{x}}' = \mathbf{\Lambda}^{-1}\tilde{\mathbf{x}}$ are known as *normalized image coordinates*: they are the coordinates that would have resulted if we had used a normalized camera. In effect, pre-multiplying by $\mathbf{\Lambda}^{-1}$ compensates for the idiosyncrasies of this particular camera.

We now note that the last of these three equations allows us to solve for the constant, λ_i so that

$$\lambda_i = r_{31}u_i + r_{32}v_i + r_{33}w_i + t_z \tag{3.29}$$

and we can now substitute this back into the first two equations to get the relations

$$\begin{bmatrix} (r_{31}u_i + r_{32}v_i + r_{33}w_i + t_z)x'_i \\ (r_{31}u_i + r_{32}v_i + r_{33}w_i + t_z)y'_i \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \\ 1 \end{bmatrix}. \tag{3.30}$$

These are two linear equations with respect to the unknown quantities $\mathbf{R}$ and $\mathbf{T}$. We can take the two equations provided by each of the I pairs of points in the world $\mathbf{w}$ and the image $\mathbf{x}$ to form

the system of equations

$$\begin{bmatrix} u_1 & v_1 & w_1 & 1 & 0 & 0 & 0 & 0 & -u_1x'_1 & -v_1x'_1 & -w_1x'_1 & -x'_1 \\ 0 & 0 & 0 & 0 & u_1 & v_1 & w_1 & 1 & -u_1y'_1 & -v_1y'_1 & -w_1y'_1 & -y'_1 \\ u_2 & v_2 & w_2 & 1 & 0 & 0 & 0 & 0 & -u_2x'_2 & -v_2x'_2 & -w_2x'_2 & -x'_2 \\ 0 & 0 & 0 & 0 & u_2 & v_2 & w_2 & 1 & -u_2y'_2 & -v_2y'_2 & -w_2y'_2 & -y'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_I & v_I & w_I & 1 & 0 & 0 & 0 & 0 & -u_Ix'_I & -v_Ix'_I & -w_Ix'_I & -x'_I \\ 0 & 0 & 0 & 0 & u_I & v_I & w_I & 1 & -u_Iy'_I & -v_Iy'_I & -w_Iy'_I & -y'_I \end{bmatrix} \begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \\ t_x \\ r_{21} \\ r_{22} \\ r_{23} \\ t_y \\ r_{31} \\ r_{32} \\ r_{33} \\ t_z \end{bmatrix} = \mathbf{0}. \tag{3.31}$$

This problem is now in the standard form $\mathbf{Ab} = \mathbf{0}$ of a *minimum direction problem*. We seek the value of $\mathbf{b}$ that minimizes $\|\mathbf{Ab}\|^2$ subject to the constraint $\|\mathbf{b}\|^2 = 1$ (to avoid the uninteresting solution $\mathbf{b} = 0$). The solution can be found by computing *singular value decomposition (SVD)* (see $Appendix$ [3]) $\mathbf{A} = \mathbf{USV}^T$ and setting $\hat{\mathbf{b}}$ to be the last column of $\mathbf{V}$ (see $Appendix$ [3]).

The estimates of $\mathbf{R}$ and $\mathbf{T}$ that we extract from $\mathbf{b}$ have had an arbitrary scale imposed on them, and we must find the correct scaling factor. This is possible because the rotation $\mathbf{R}$ has a pre-defined scale (its rows and columns must all have norm one). In practice, we first find the closest true rotation matrix to $\mathbf{R}$ which also forces our estimate to be a valid orthogonal matrix. This is an instance of the *orthogonal Procrustes problem* (see $Appendix$ [3]). The solution is found by computing the singular value decomposition $\mathbf{R} = \mathbf{USV^T}$ and setting $\hat{\mathbf{R}} = \mathbf{UV^T}$. Now, we re-scale the translation $\mathbf{T}$. The scaling factor can be estimated by taking the average ratio of the nine entries of our initial estimate of $\mathbf{R}$ to the final one $\hat{\mathbf{R}}$ so that

$$\hat{\mathbf{T}} = \sum_{m=1}^{3}\sum_{n=1}^{3}\frac{\hat{R}_{mn}}{R_{mn}}\mathbf{T}. \tag{3.32}$$

Finally, we must check that the sign of t_z is positive, indicating that the object is in front of the camera. If this is not the case then be multiply both $\hat{\mathbf{T}}$ and $\hat{\mathbf{R}}$ by minus one.

This scrappy algorithm is typical of methods that use homogeneous coordinates. The resulting estimates $\hat{\mathbf{T}}$and $\hat{\mathbf{R}}$ can be quite inaccurate in the presence of noise in the measured image positions. However, they usually suffice as a reasonable starting point for the subsequent nonlinear optimization of the true objective function (equation 3.26) for this problem. That optimization must be carried out while ensuring that $\mathbf{R}$ remains a valid rotation matrix.

Note that this algorithm requires a minimum of 11 equations to solve the minimum direction problem. Since each point contributes two equations, this means we require $I = 6$ points for a unique solution. However, there are only really six unknowns (the rotation and translation in

3D) and so a minimal solution would require only $I = 3$ points.

Chapter 4

EPIPOLAR GEOMETRY

4.1 Two-view geometry

In this chapter, we show that there is a geometric relationship between corresponding points in two images of the same scene. This relationship depends only on the intrinsic parameters of the two cameras and their relative translation and rotation.

4.1.1 The epipolar constraint

Consider a single camera viewing a 3D point $\mathbf{w}$ in the world. We know that $\mathbf{w}$ must lie somewhere on the ray that passes through the optical center and position $\mathbf{x_1}$ on the image plane (figure 4.1). However, from one camera alone, we cannot know how far along this ray the point is.

Now consider a second camera viewing the same 3D world point. We know from the first camera that this point must lie along a particular ray in 3D space. It follows that the projected position $\mathbf{x_2}$ of this point in the second image must lie somewhere along the projection of this ray in the second image. The ray in 3D projects to a 2D line which is known as an *epipolar line*.

This geometric relationship tells us something important: for any point in the first image, the corresponding point in the second image is constrained to lie on a line. This is known

as the *epipolar constraint*. The particular line that it is constrained to lie on depends on the intrinsic parameters of the cameras and the relative translation and rotation of the two cameras (determined by the extrinsic parameters).

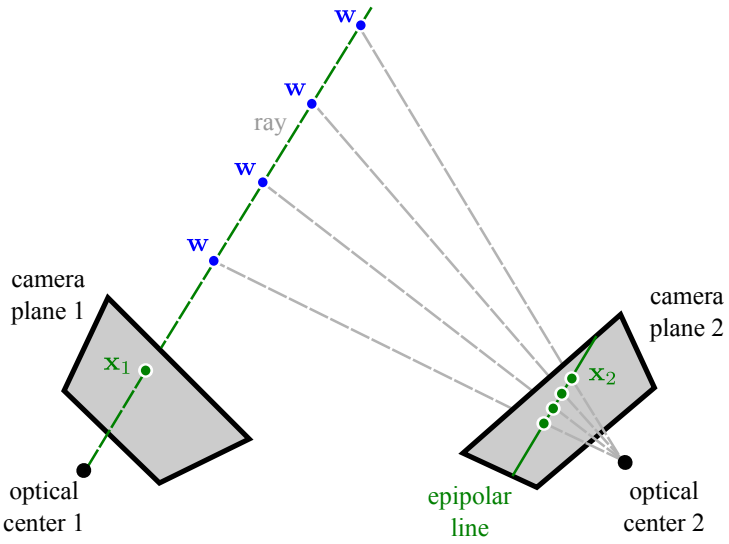


Figure 4.1: Epipolar line. Consider point $\mathbf{x_1}$ in the first image. The 3D point $\mathbf{w}$ that projected to $\mathbf{x_1}$ must lie somewhere along the ray that passes from the optical center of camera 1 through the position $\mathbf{x_1}$ in the image plane (dashed green line). However, we don't know where along that ray it lies (4 possibilities shown). It follows that $\mathbf{x_2}$, the projected position in camera 2 must lie somewhere on the projection of this ray. The projection of this ray is a line in image 2 and is referred to as an epipolar line. Figure from [3].

The epipolar constraint has two important practical implications.

1. Given the intrinsic and extrinsic parameters, we can find point correspondences relatively easily: for a given point in the first image, we only need to perform a 1D search along the epipolar line in the second image for the corresponding position.

2. The constraint on corresponding points is a function of the intrinsic and extrinsic parameters; given the intrinsic parameters, we can use the observed pattern of point correspondences to determine the extrinsic parameters and hence establish the geometric relationship between the two cameras.

4.1.2 Epipoles

Now consider a number of points in the first image. Each is associated with a ray in 3D space. Each ray projects to form an epipolar line in the second image. Since all the rays converge at the optical center of the first camera, the epipolar lines must converge at a single point in the second image plane; this is the image in the second camera of the optical center of the first camera and is known as the *epipole* (figure 4.2). Similarly, points in image 2 induce epipolar lines in image 1, and these epipolar lines converge at the epipole in image 1. This is the image in camera 1 of the optical center of camera 2.

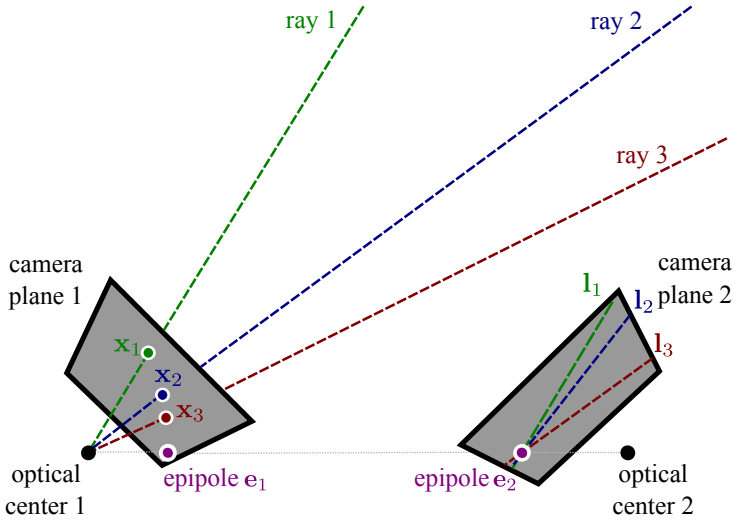


Figure 4.2: Epipoles. Consider several observed points $\{\mathbf{x_i}\}_{i=1}^{I}$ in image 1. For each point, the corresponding 3D world position $\mathbf{w_i}$ lies on a different ray. Each ray projects to an epipolar line $\mathbf{l_i}$ in image 2. Since the rays converge in 3D space at the optical center of camera 1, the epipolar lines must also converge. The point where they converge is known as the epipole $\mathbf{e_2}$. It is the projection of the optical center of camera 1 into camera 2. Similarly, the epipole $\mathbf{e_1}$ is the projection of the optical center of camera 2 into camera 1. The figure was reproduced from [3]

The epipoles are not necessarily within the observed images: the epipolar lines may converge to a point outside the visible area. Two common cases are illustrated in figure 4.3. When the cameras are oriented in the same direction (i.e., no relative rotation) and the translation is perpendicular to their optical axes (figure 4.3a) then the epipolar lines are parallel and the epipoles (where they converge) are hence at infinity. When the cameras are oriented in the same direction and the translation is parallel to their optical axes (figure 4.3b), then the epipoles are in the middle of the images and the epipolar lines form a radial pattern. These examples illustrate that the pattern of epipolar lines provides information about the relative position and orientation of the cameras.

4.2 The essential matrix

Now we will capture these geometric intuitions in the form of a mathematical model. For simplicity, we will assume that the world coordinate system is centered on the first camera so that the extrinsic parameters (rotation and translation) of the first camera are $\{\mathbf{I}, \mathbf{0}\}$. The second camera may be in any general position $\{\mathbf{R}, \mathbf{T}\}$. We will further assume that the cameras are normalized so that $\mathbf{\Lambda_1} = \mathbf{\Lambda_2} = \mathbf{I}$. In homogeneous coordinates, a 3D point $\mathbf{w}$ is projected into the two cameras as

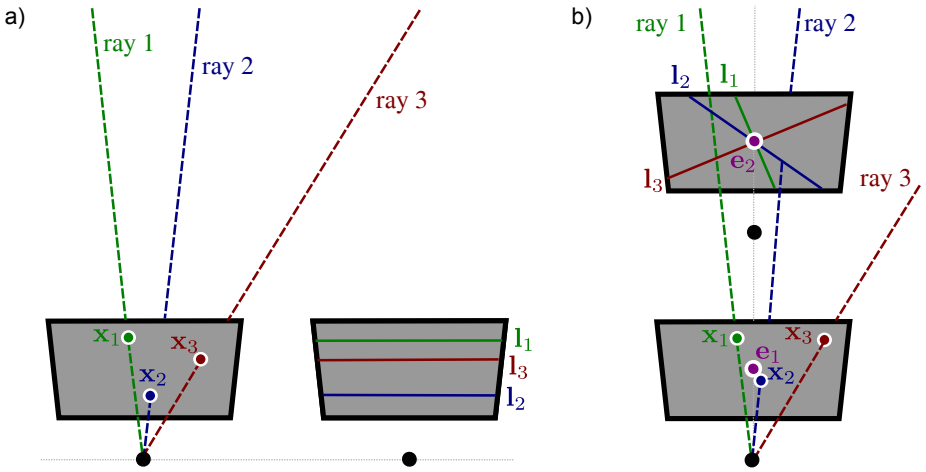


Figure 4.3: Epipolar lines and epipoles. a) When the camera movement is a pure translation perpendicular to the optical axis (parallel to the image plane) the epipolar lines are parallel and the epipole is at infinity. b) When the camera movement is a pure translation along the optical axis the epipoles are in the center of the image and the epipolar lines form a radial pattern. The figure was copied from [3].

$$\begin{aligned} \lambda\tilde{\mathbf{x}}_1 &= [\mathbf{I}, \mathbf{0}]\tilde{\mathbf{w}} \\ \lambda\tilde{\mathbf{x}}_1 &= [\mathbf{R}, \mathbf{T}]\tilde{\mathbf{w}}. \end{aligned} \tag{4.1}$$

where $\tilde{\mathbf{x}}_1$ is the observed position in the first camera, $\tilde{\mathbf{x}}_2$ is the observed position in the second camera, and both are expressed in homogeneous coordinates.

Expanding the first of these relations we get

$$\lambda_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \tag{4.2}$$

This simplifies to

$$\lambda_1\tilde{\mathbf{x}}_1 = \mathbf{w}. \tag{4.3}$$

By a similar process, the projection in the second camera can be written as

$$\lambda_2\tilde{\mathbf{x}}_2 = \mathbf{R}\mathbf{w} + \mathbf{T}. \tag{4.4}$$

Finally, substituting equation 4.3 into equation 4.4 yields

$$\lambda_2\tilde{\mathbf{x}}_2 = \lambda_1\mathbf{R}\tilde{\mathbf{x}}_1 + \mathbf{T}. \tag{4.5}$$

This relationship represents a constraint between the possible positions of corresponding points

$\mathbf{x}_1$ and $\mathbf{x}_2$ in the two images. The constraint is parameterized by the rotation and translation $\{\mathbf{R}, \mathbf{T}\}$ of the second camera relative to the first.

We will now manipulate the relationship in equation 4.5 into a form that can be more easily related to the epipolar lines and the epipoles. We first take the cross product of both sides with the translation vector t. This removes the last term as the cross product of any vector with itself is zero. Now we have

$$\lambda_2 \mathbf{T} \times \tilde{\mathbf{x}}_2 = \lambda_1 \mathbf{T} \times \mathbf{R}\tilde{\mathbf{x}}_1. \tag{4.6}$$

Then we take the inner product of both sides with $\tilde{\mathbf{x}}_2$. The left hand side disappears since $\mathbf{T} \times \tilde{\mathbf{x}}_2$ must be perpendicular to $\tilde{\mathbf{x}}_2$, and so we have

$$\tilde{\mathbf{x}}_2^T \mathbf{T} \times \mathbf{R}\tilde{\mathbf{x}}_1 = 0, \tag{4.7}$$

where we have also eliminated the scaling factors λ_1 and λ_2 by dividing by them. Finally, we note that the cross product operation $\mathbf{T}\times$ can be expressed as multiplication by the rank 2 skew-symmetric 3×3 matrix $\mathbf{T}_\times$:

$$\mathbf{T}_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}. \tag{4.8}$$

Hence equation 4.7 has the form

$$\tilde{\mathbf{x}}_2^T \mathbf{E} \tilde{\mathbf{x}}_1 = 0, \tag{4.9}$$

where $\mathbf{E} = \mathbf{T}_\times \mathbf{R}$ is known as the essential matrix. Equation 4.9 is an elegant formulation of the mathematical constraint between the positions of corresponding points $\mathbf{x}_1$ and $\mathbf{x}_2$ in two normalized cameras.

4.2.1 Properties of the essential matrix

The 3×3 essential matrix captures the geometric relationship between the two cameras and has rank 2 so that $det[\mathbf{E}] = 0$. The first two *singular values* of the essential matrix are always identical and the third is zero. It depends only on the rotation and translation between the cameras, each of which has 3 parameters, and so one might think it would have 6 degrees of freedom. However, it operates on homogeneous variables $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$ and is hence ambiguous up to scale: multiplying all of the entries of the essential matrix by any constant does not change its properties. For this reason, it is usually considered as having 5 degrees of freedom [4].

Since there are fewer degrees of freedom than there are unknowns, the nine entries of the matrix must obey a set of algebraic constraints. These can be expressed compactly as

$$2\mathbf{E}\mathbf{E}^T\mathbf{E} - trace[\mathbf{E}\mathbf{E}^T]\mathbf{E} = \mathbf{0}. \tag{4.10}$$

These constraints are sometimes exploited in the computation of the essential matrix, although in this volume we use a simpler method (section 4.4).

The epipolar lines are easily retrieved from the essential matrix. The condition for a point being on a line is $ax + by + c = 0$ or

$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0. \tag{4.11}$$

In homogeneous co-ordinates, this can be written as $\mathbf{l}\tilde{\mathbf{x}} = 0$ where $\mathbf{l} = [a, b, c]$ is a 1×3 vector representing the line.

Now consider the essential matrix relation

$$\tilde{\mathbf{x}}_2^T \mathbf{E} \tilde{\mathbf{x}}_1 = 0, \tag{4.12}$$

Since $\tilde{\mathbf{x}}_2^T \mathbf{E}$ is a 1×3 vector, this relationship has the form $\mathbf{l_1}\tilde{\mathbf{x}}_1 = 0$. The line $\mathbf{l_1} = \tilde{\mathbf{x}}_2^T \mathbf{E}$ is the epipolar line in image 1 due to the point $\mathbf{x}_2$ in image 2. By a similar argument, we can find the epipolar line $\mathbf{l_2}$ in the second camera due to the point $\mathbf{x}_1$ in the first camera. The final relations are

$$\begin{aligned} \mathbf{l_1} &= \tilde{\mathbf{x}}_2^T \mathbf{E} \\ \mathbf{l_2} &= \tilde{\mathbf{x}}_1^T \mathbf{E}^T. \end{aligned} \tag{4.13}$$

The epipoles can also be extracted from the essential matrix. Every epipolar line in image 1 passes through the epipole $\tilde{\mathbf{e}}_1$, so at the epipole $\tilde{\mathbf{e}}_1$ we have $\tilde{\mathbf{x}}_2^T \mathbf{E} \tilde{\mathbf{e}}_1 = 0$ for all $\tilde{\mathbf{x}}_2$. This implies that $\tilde{\mathbf{e}}_1$ must lie in the right null-space of $\mathbf{E}$ (see *Appendix* [3]). By a similar argument, the epipole $\tilde{\mathbf{e}}_2$ in the second image must lie in the left null space of $\mathbf{E}$. Hence, we have the relations

$$\begin{aligned} \tilde{\mathbf{e}}_1 &= \mathbf{null}[\mathbf{E}] \\ \tilde{\mathbf{e}}_2 &= \mathbf{null}[\mathbf{E}^T]. \end{aligned} \tag{4.14}$$

In practice, the two epipole points can be retrieved by computing the singular value decomposition $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$ of the essential matrix, and setting $\tilde{\mathbf{e}}_1$ to the last column of $\mathbf{V}$ and $\tilde{\mathbf{e}}_2$ to the last row of $\mathbf{U}$.

4.2.2 Decomposition of the essential matrix

We saw previously that the essential matrix is defined as

$$\mathbf{E} = \mathbf{T}_\times \mathbf{R}, \tag{4.15}$$

where $\mathbf{R}$ and $\mathbf{T}$ are the rotation matrix and translation vector that map points in the coordinate system of camera 2 to the coordinate system of camera 1, and $\mathbf{T}_\times$ is a 3×3 matrix derived from the translation vector.

We will defer the question of how to compute the essential matrix from a set of corresponding points until section 4.3. For now, we will concentrate on how to decompose a given essential matrix $\mathbf{E}$ to recover this rotation $\mathbf{R}$ and translation $\mathbf{T}$. This is known as the *relative orientation* problem.

In due course, we shall see that we can compute the rotation exactly, whereas it is only possible to compute the translation up to an unknown scaling factor. This remaining uncertainty reflects the geometric ambiguity of the system; from the images alone, we cannot tell if these cameras are far apart and looking at a large distant object or close together and looking at a small nearby object.

To decompose $\mathbf{E}$, we define the matrix

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.16}$$

and then take the singular value decomposition $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$. We now choose

$$\begin{aligned} \mathbf{T}_\times &= \mathbf{U}\mathbf{L}\mathbf{W}\mathbf{U}^T \\ \mathbf{R} &= \mathbf{U}\mathbf{W}^{-1}\mathbf{V}^T. \end{aligned} \tag{4.17}$$

It is convention to set the magnitude of the translation vector $\mathbf{T}$ that is recovered from the matrix $\mathbf{T}_\times$ to unity. The above decomposition is not obvious, but it is easily checked that multiplying the derived expressions for $\mathbf{T}_\times$ and $\mathbf{R}$ yields $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$. This method assumes that we started with a valid essential matrix where the first two singular values are identical and the third is zero. If this is not the case (due to noise) we can substitute $\mathbf{L}' = \mathbf{diag}[1, 1, 0]$ for $\mathbf{L}$ in the solution for $\mathbf{T}_\times$ [4].

This solution is only one of four possible combinations of $\mathbf{R}$ and $\mathbf{T}$ that are compatible with $\mathbf{E}$ (figure 4.4). This four-fold ambiguity is due to the fact that the pinhole model cannot distinguish between objects that are behind the camera (and are not imaged in real cameras) and those that are in front of the camera. Part of the uncertainty is captured mathematically by our lack of knowledge of the sign of the essential matrix (recall it is ambiguous up to scale) and hence the sign of the recovered translation. Hence, we can generate a second solution by multiplying the translation vector by -1. The other component of the uncertainty results from an ambiguity in the decomposition of the essential matrix; we can equivalently replace $\mathbf{W}$ for $\mathbf{W}^{-1}$ in the decomposition procedure, and this leads to two more solutions.

Fortunately, we can resolve this ambiguity using a corresponding pair of points from the two images and the Triangulation algorithm which is developed in the next section.

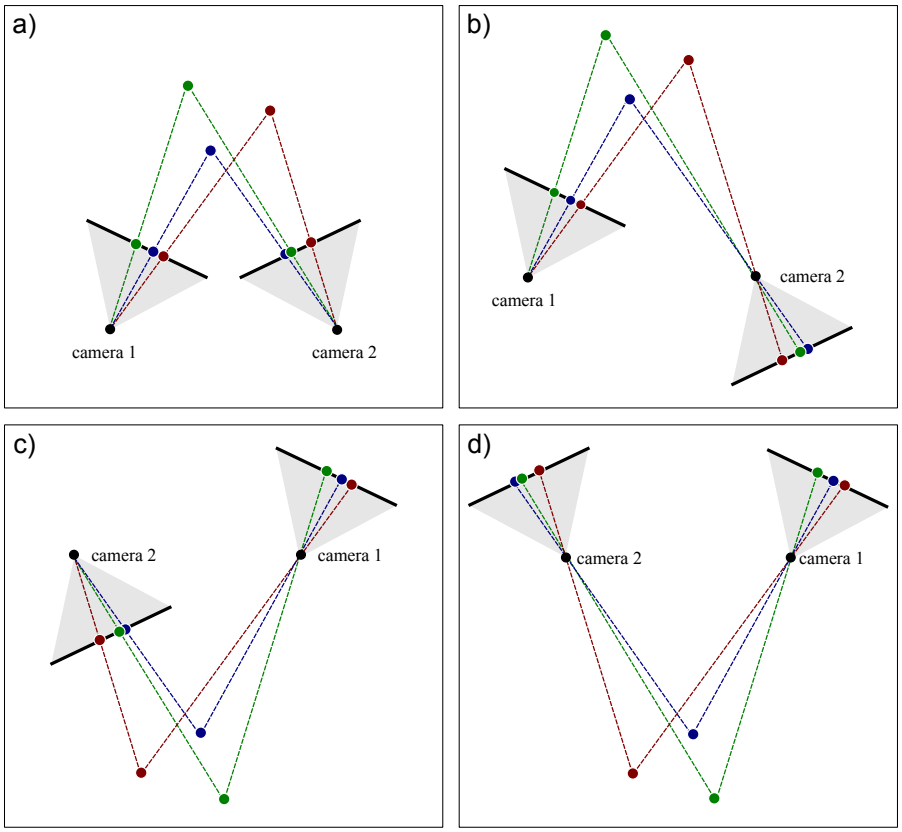


Figure 4.4: Four-fold ambiguity of reconstruction from two pinhole cameras. The mathematical model for the pinhole camera does not distinguish between points that are in front of and points that are behind the camera. This leads to a four-fold ambiguity when we extract the rotation **R** and translation **T** relating the cameras from the essential matrix. a) Correct solution. Points are in front of both cameras. b) Incorrect solution. The images are identical, but with this interpretation, the points are behind camera 2. c) Incorrect solution with points behind camera 1. d) Incorrect solution with points behind both cameras. The figure was copied from [3].

4.3 Triangulation

As shown a world point, projected into the pair of corresponding points $\mathbf{x}_l$ and $\mathbf{x}_r$, lies at the intersection of the two rays O_l through $\mathbf{x}_l$ and from O_r through $\mathbf{x}_r$ respectively. In our assumptions, the rays are known and the intersection can be computed. The problem is, since parameters and image locations are known only approximately, *the two rays will not actually intersect in space*; their intersection can only be estimated as the point of minimum distance from both rays. This is what we set off to do.

Let $a\mathbf{x}_l$ $(a \in \mathbb{R})$ be the ray, l, through O_l and $\mathbf{x}_l$. Let $\mathbf{T}+b\mathbf{R}^T\mathbf{x}_r$ $(b \in \mathbb{R})$ be the ray, r, through O_r and $\mathbf{x}_r$ expressed in the left reference frame. Let $\mathbf{w}$ be a vector orthogonal to both l and r. Our problem reduces to determining the midpoint, P', of the segment parallel to $\mathbf{w}$ that joins l and r (Figure 4.5).

This is very simple because the endpoints of the segment, say $a_0\mathbf{x}_l$ and $\mathbf{T} + b_0\mathbf{R}^T\mathbf{x}_r$, can be computed solving the linear system of equations

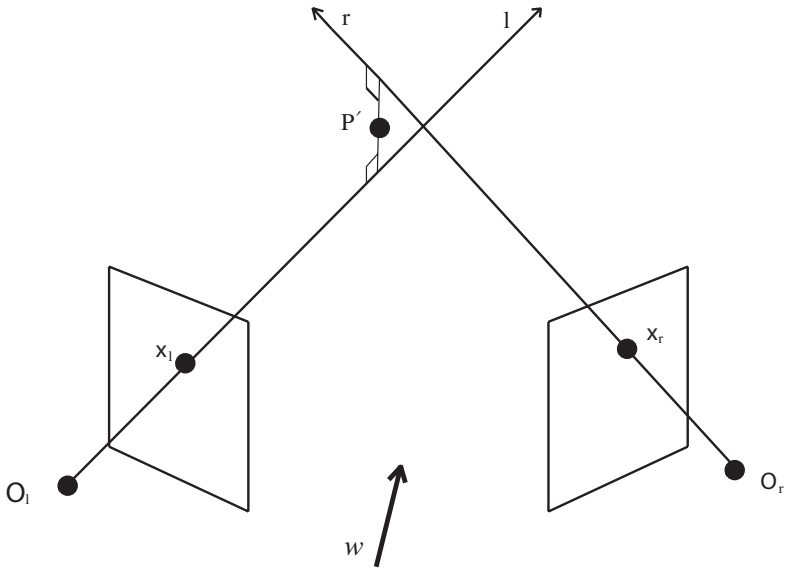


Figure 4.5: Triangulation with nonintersecting rays. The figure was reproduced from [5].

$$a\mathbf{x}_l - b\mathbf{R}^T\mathbf{x}_r + c(\mathbf{x}_l \times \mathbf{R}^T\mathbf{x}_r) = \mathbf{T} \tag{4.18}$$

for $a_0, b_0,$ and c_0. We summarize this simple method in the Algorithm 1 below. All vectors and coordinates are referred to the left camera reference frame. The input is formed by a set of corresponding points; let $\mathbf{x}_l$ and $\mathbf{x}_r$ be a generic pair.

Algorithm 1 Triangulation

Let $a\mathbf{x}_l,\ a \in \mathbb{R}$ be the ray, l, through $O_l\ (a = 0)$ and $\mathbf{x}_l\ (a = 1)$. Let $\mathbf{T} + b\mathbf{R}^T\mathbf{x}_r\ b, \in \mathbb{R}$ be the ray, r, through $O_r\ (b = 0)$ and $\mathbf{x}_r\ (b = 1)$. Let $\mathbf{w} = \mathbf{x}_l \times \mathbf{R}^T\mathbf{x}_r$ the vector orthogonal to both l and r, and $a\mathbf{x}_r + c\mathbf{w}, c \in \mathbb{R}$, the line w through $a\mathbf{x}_l$ (for some fixed a) and the parallel to $\mathbf{w}$.

1. Determine the endpoints of the segment, s, belonging to the line parallel to $\mathbf{w}$ that joins l and r, $a_0\mathbf{x}_l$ and $\mathbf{T} + b_0\mathbf{R}^T\mathbf{x}_r$, by solving equation 4.18.
2. The trinagulated point, P', is the midpoint of the segment s.
   The output is the set of reconstructed 3-D points.

4.4 The fundamental matrix

The derivation of the essential matrix in section 4.2 used normalized cameras (where $\mathbf{\Lambda}_1 = \mathbf{\Lambda}_2 = \mathbf{I}$). The fundamental matrix plays the role of the essential matrix for cameras with arbitrary intrinsic matrices $\mathbf{\Lambda}_1$ and $\mathbf{\Lambda}_2$. The general projection equations for the two cameras are

$$\begin{aligned} \lambda_1\tilde{\mathbf{x}}_\mathbf{1} &= \mathbf{\Lambda}_1[\mathbf{I}, \mathbf{0}]\tilde{\mathbf{w}} \\ \lambda_2\tilde{\mathbf{x}}_\mathbf{2} &= \mathbf{\Lambda}_2[\mathbf{R}, \mathbf{T}]\tilde{\mathbf{w}}, \end{aligned} \tag{4.19}$$

and we can use similar manipulations to those presented in section 4.2 to derive the constraint

$$\tilde{\mathbf{x}}_2^T \boldsymbol{\Lambda}_2^{-T} \mathbf{E} \boldsymbol{\Lambda}_1^{-1} \tilde{\mathbf{x}}_2 = 0, \tag{4.20}$$

or

$$\tilde{\mathbf{x}}_2^T \mathbf{F} \tilde{\mathbf{x}}_2 = 0, \tag{4.21}$$

where the 3×3 matrix $\mathbf{F} = \boldsymbol{\Lambda}_2^{-T} \mathbf{E} \boldsymbol{\Lambda}_1^{-1} = \boldsymbol{\Lambda}_2^{-T} \mathbf{T}_\times \mathbf{R} \boldsymbol{\Lambda}_1^{-1}$ is termed the *fundamental matrix* [29]. Like the essential matrix, it also has rank two, but unlike the essential matrix it has seven degrees of freedom.

If we know the fundamental matrix $\mathbf{F}$ and the intrinsic matrices $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$, it is possible to recover the essential matrix $\mathbf{E}$ using the relation

$$\mathbf{E} = \boldsymbol{\Lambda}_2^T \mathbf{F} \boldsymbol{\Lambda}_1, \tag{4.22}$$

and this can further be decomposed to find the rotation and translation between the cameras using the method of Section 4.2.2. It follows that for calibrated cameras, if we can estimate the fundamental matrix, then we can find the rotation and translation between the cameras. Hence, we now turn our attention on how to compute the fundamental matrix.

4.4.1 Estimation of the fundamental matrix

The fundamental matrix relation (equation 4.21) is a constraint on the possible positions of corresponding points in the first and second images. This constraint is parameterized by the nine entries of $\mathbf{F}$. It follows that if we analyze a set of corresponding points, we can observe *how* they are constrained, and from this we can deduce the entries of the fundamental matrix $\mathbf{F}$.

Figure 4.6: The cost function is the sum of the squares of the distances between these epipolar lines and the points (yellow arrows). This is termed *symmetric epipolar distance*. Figure from [3].

A suitable cost function for the fundamental matrix can be found by considering the epipolar lines. Consider a pair of matching points $\mathbf{x}_{i1}, \mathbf{x}_{i2}$ in images 1 and 2, respectively. Each point

induces an epipolar line in the other image: the point $\mathbf{x}_{i1}$ induces line $\mathbf{l}_{i2}$ in image 2 and the point $\mathbf{x}_{i2}$ induces the line $\mathbf{l}_{i1}$ in image 1. When the fundamental matrix is correct, each point should lie exactly on the epipolar line induced by the corresponding point in the other image (figure 4.6). We hence minimize the squared distance between every point and the epipolar line predicted by its match in the other image so that

$$\hat{\mathbf{F}} = \arg\min_{\mathbf{F}} \left[\sum_{i=1}^{I} \left((dist[\mathbf{x}_{i1}, \mathbf{l}_{i1}])^2 + (dist[\mathbf{x}_{i2}, \mathbf{l}_{i2}])^2 \right) \right], \tag{4.23}$$

where the distance between a 2D point $\mathbf{x} = [x, y]^T$ and a line $\mathbf{l} = [a, b, c]$ is

$$dist[\mathbf{x}, \mathbf{l}] = \frac{ax + by + c}{\sqrt{a^2 + b^2}}. \tag{4.24}$$

Here too, it is not possible to find the minimum of equation 4.23 in closed form, and we must rely on nonlinear optimization methods. It is possible to get a good starting point for this optimization using the *eight-point algorithm*.

4.4.2 The eight-point algorithm

The eight-point algorithm converts the corresponding 2D points to homogeneous coordinates and then solves for the fundamental matrix in closed form. It does not directly optimize the cost function in equation 4.23, but instead minimizes an algebraic error. However, the solution to this problem is usually very close to the values that optimize the desired cost function.

In homogeneous coordinates, the relationship between the i^{th} point $\mathbf{x}_{i1} = [x_{i1}, y_{i1}]^T$ in image 1 and the i^{th} point $\mathbf{x}_{i2} = [x_{i2}, y_{i2}]^T$ in image 2 is

$$\begin{bmatrix} x_{i2} & y_{i2} & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{31} & f_{33} \end{bmatrix} \begin{bmatrix} x_{i1} \\ y_{i1} \\ 1 \end{bmatrix} = 0, \tag{4.25}$$

where f_{pq} represents one of the entries in the fundamental matrix. When we write this constraint out in full, we get

$$x_{i2}x_{i1}f_{11} + x_{i2}y_{i1}f_{12} + x_{i2}f_{13} + y_{i2}x_{i1}f_{21} + y_{i2}y_{i1}f_{22} + y_{i2}f_{23} + x_{i1}f_{31} + y_{i1}f_{32} + f_{33} = 0. \tag{4.26}$$

This can be expressed as an inner product

$$\begin{bmatrix} x_{i2}x_{i1}, x_{i2}y_{i1}, x_{i2}, y_{i2}x_{i1}, y_{i2}y_{i1}, y_{i2}, x_{i1}y_{i1}, 1 \end{bmatrix} \mathbf{f} = 0, \tag{4.27}$$

where $\mathbf{f} = [f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}]^T$ is a vectorized version of the fundamental matrix, $\mathbf{F}$.

This provides one linear constraint on the elements of $\mathbf{F}$. Consequently, given I matching points, we can stack these constraints to form the system

$$\mathbf{Af} = \begin{bmatrix} x_{12}x_{11} & x_{12}y_{11} & x_{12} & y_{12}x_{11} & y_{12}y_{11} & y_{12} & x_{11} & y_{11} & 1 \\ x_{22}x_{21} & x_{22}y_{21} & x_{22} & y_{22}x_{21} & y_{22}y_{21} & y_{22} & x_{21} & y_{21} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{I2}x_{I1} & x_{I2}y_{I1} & x_{I2} & y_{I2}x_{I1} & y_{I2}y_{I1} & y_{I2} & x_{I1} & y_{I1} & 1 \end{bmatrix} \mathbf{f} = 0. \tag{4.28}$$

Since the elements of $\mathbf{f}$ are ambiguous up to scale, we solve this system with the constraint that $\|f\| = 1$. This also avoids the trivial solution $\mathbf{f} = \mathbf{0}$. This is a minimum direction problem (see *Appendix* [3]). The solution can be found by taking the singular value decomposition, $\mathbf{A} = \mathbf{USV^T}$ and setting $\mathbf{f}$ to be the last column of $\mathbf{V}$. The matrix $\mathbf{F}$ is then formed by reshaping $\mathbf{f}$ to form a 3×3 matrix [4].

Algorithm 2 Eight-point algorithm for Fundamental matrix

Input : Point pairs $\{\mathbf{x}_{1i}, \mathbf{x}_{2i}\}_{i=1}^{I}$
Output: Fundamental matrix $\mathbf{F}$

// Compute statistics of data
$\boldsymbol{\mu}_1 = \sum_{i=1}^{I} \mathbf{x}_{1i}/I$
$\boldsymbol{\Sigma}_1 = \sum_{i=1}^{I} (\mathbf{x}_{1i} - \boldsymbol{\mu}_1)(\mathbf{x}_{1i} - \boldsymbol{\mu}_1)/I$
$\boldsymbol{\mu}_2 = \sum_{i=1}^{I} \mathbf{x}_{2i}/I$
$\boldsymbol{\Sigma}_2 = \sum_{i=1}^{I} (\mathbf{x}_{2i} - \boldsymbol{\mu}_2)(\mathbf{x}_{2i} - \boldsymbol{\mu}_2)/I$
for $k = 1$ **to** K **do**
  // Compute transformed coordinates
  $\mathbf{x}_{1i} = \boldsymbol{\Sigma}_1^{-1/2}(\mathbf{x}_{1i} - \boldsymbol{\mu}_1)$
  $\mathbf{x}_{2i} = \boldsymbol{\Sigma}_2^{-1/2}(\mathbf{x}_{2i} - \boldsymbol{\mu}_2)$
  // Compute constraint
  $\mathbf{A}_i = [x_{i2}x_{i1}, x_{i2}y_{i1}, x_{i2}, y_{i2}x_{i1}, y_{i2}y_{i1}, y_{i2}, x_{i1}, y_{i1}, 1]$
end for
// Append constraints and solve
$\mathbf{A} = [\mathbf{A_1}; \mathbf{A_2}; ... \mathbf{A_I}]$
$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \mathbf{svd}[\mathbf{A}]$
$\mathbf{F} = [v_{19}, v_{29}, v_{39}; v_{49}, v_{59}, v_{69}; v_{79}, v_{89}, v_{99}]$
// Compensate for transformation
$T1 = [\boldsymbol{\Sigma}_1^{-1/2}, \boldsymbol{\Sigma}_1^{-1/2}\boldsymbol{\mu}_1; 0, 0, 1]$
$T2 = [\boldsymbol{\Sigma}_2^{-1/2}, \boldsymbol{\Sigma}_2^{-1/2}\boldsymbol{\mu}_2; 0, 0, 1]$
$\mathbf{F} = \mathbf{T_2^T F T_1}$
// Ensure that matrix has rank 2
$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \mathbf{svd}[\mathbf{F}]$
$\sigma_{33} = 0$
$\mathbf{F} = \mathbf{USV}^T$

There are 8 degrees of freedom in the fundamental matrix (it is ambiguous with respect to scale) and so we require a *minimum* of $I = 8$ pairs of points. For this reason, this algorithm

is called the *eight-point algorithm*. This algorithm takes a set of $I \geq 8$ point correspondences $\mathbf{x}_{i1}, \mathbf{x}_{i2}{}_{i=1}^{I}$ between two images and computes the fundamental matrix using the 8 point algorithm. To improve the numerical stability of the algorithm, the point positions are transformed to have unit mean and spherical covariance before the calculation proceeds. The resulting fundamental matrix is modified to compensate for this transformation. This algorithm 2 is usually used to compute an initial estimate for a subsequent non-linear optimization of the symmetric epipolar distance.

In practice, there are several further concerns in implementing this algorithm:

- Since the data are noisy, the singularity constraint of the resulting fundamental matrix will not be obeyed in general (i.e., the estimated matrix will be full rank, not rank two). We re-introduce this constraint by taking the singular decomposition of $\mathbf{F}$, setting the last singular value to zero, and multiplying the terms back out. This provides the closest singular matrix under a Frobenius norm.

- Equation 4.28 is badly scaled since some terms are on the order of pixels squared ($\sim$ 10000) and some are of the order ~ 1. To improve the quality of the solution, it is wise to pre-normalize the data. We transform the points in image 1 as $\tilde{x}'_{i1} = \mathbf{T_1}\tilde{x}_{i1}$ and the points in image 2 as $\tilde{x}'_{i2} = \mathbf{T_2}\tilde{x}_{i2}$. The transformations $\mathbf{T_1}$ and $\mathbf{T_2}$ are chosen to map the mean of the points in their respective image to zero, and to ensure that the variance in the x- and y-dimensions is one. We then compute the matrix $\mathbf{F}'$ from the transformed data using the eight-point algorithm, and recover the original fundamental matrix as $\mathbf{F} = \mathbf{T}_2^T\mathbf{F}'\mathbf{T_1}$.

- The algorithm will only work if the three-dimensional positions $\mathbf{w_i}$ corresponding to the eight pairs of points $\mathbf{x}_{i1}, \mathbf{x}_{i2}$ are in general position. For example, if they all fall on a plane, then the equations become degenerate and we cannot get a unique solution; here, the relation between the points in the two images is given by a homography (see chapter *Homography*). Similarly, in the case where there is no translation (i.e., $t = 0$), the relation between the two images is a homography and there is no unique solution for the fundamental matrix.

- In the subsequent nonlinear optimization, we must also ensure that the rank of $\mathbf{F}$ is two. In order to do this, it is usual to re-parameterize the fundamental matrix to ensure that this will be the case.

4.4.3 Robust computation of fundamental matrix with RANSAC

The goal of this algorithm is to estimate the fundamental matrix from 2D point pairs $\{\mathbf{x}_{i1}, \mathbf{x}_{i2}\}_{i=1}^{I}$ to another in the case where some of the point matches are known to be wrong (outliers). The robustness is achieved by applying the *Random sample consensus* (RANSAC) algorithm. Since the fundamental matrix has a eight unknown quantities, we randomly select eight point pairs at

each stage of the algorithm (each pair contributes one constraint). The goal of RANSAC is to identify which points are outliers and to eliminate them from the final fit.

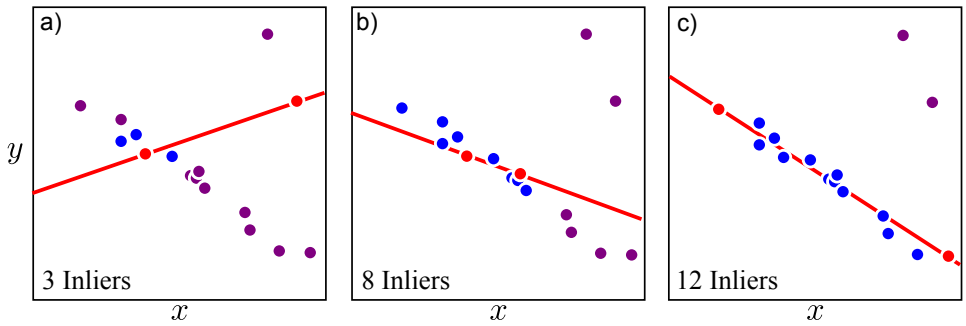


Figure 4.7: RANSAC procedure. a) We select a random minimal subset of points to fit the line (red points). We fit the line to these points and count how many of the other points agree with this solution (blue points). These are termed inliers. Here there are only three inliers. b,c) This procedure is repeated with different minimal subsets of points. After a number of iterations, we choose the fit that had the most inliers. We refit the line using only the inliers from this fit.

RANSAC works by repeatedly fitting models based on random subsets of the data. The hope is that sooner or later, there will be no outliers in the chosen subset, and so we will fit a good model. To enhance the probability of this happening, RANSAC chooses subsets of the minimal size required to uniquely fit the model. For example, in the case of the line, it would choose subsets of size two. We repeat this procedure a number of times: on each iteration we choose a random minimal subset of points, fit a model, and count the number of data points that agree (the inliers). After a predetermined number of iterations, we then choose the model that had the most inliers and refit the model from these alone. The complete RANSAC algorithm hence proceeds as follows (figure 4.7)

Algorithm 3 RANSAC: Robust fit of a model to a data set S which contains outliers.

---

1. Randomly select a sample of s data points from S and instantiate the model from this subset.
2. Determine the set of data points S_i which are within a distance threshold t of the model. The set S_1, is the consensus set of the sample and defines the inliers of S.
3. If the size of S_i (the number of inliers) is greater than some threshold T, re-estimate the model using all the points in S_i and terminate.
4. If the size of S_i is less than T, select a new subset and repeat the above,
5. After N trials the largest consensus set S_i is selected, and the model is re-estimated using all the points in the subset S_i.

4.5 The Gold Standard method

The Maximum Likelihood (ML) estimate of the fundamental matrix depends on the assumption of an error model. We make the assumption that noise in image point measurements obeys a Gaussian distribution. In that case the ML estimate is the one that minimizes the geometric

distance (which is reprojection error)

$$\sum_i d(\mathbf{x}_{i2}, \mathbf{x}_{i1})^2 + d(\mathbf{x'}_{i2}, \mathbf{x'}_{i1})^2 \tag{4.29}$$

where $\mathbf{x}_i \leftrightarrow \mathbf{x'}_i$ are the measured correspondence, and $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}'_i$ are the estimated "true" correspondences that satisfy $\hat{\mathbf{x}}_i'^T \mathbf{F} \hat{\mathbf{x}}_i = 0$ exactly for some rank-2 matrix $\mathbf{F}$, the estimated fundamental matrix [4].

Algorithm 4 The Gold Standard algorithm for estimating F from image correspondences.

1. Make an initial estimate of $\hat{\mathbf{F}}$ using the normalized 8-point algorithm.
2. From this $\hat{\mathbf{F}}$ extract two camera matrices $\mathbf{P} = [\mathbf{I}, \mathbf{0}]$ and $\mathbf{P'} = [[\mathbf{e'}]_\times \hat{\mathbf{F}}, \mathbf{e'}]$ where $\mathbf{e'}$ obtained from $\hat{\mathbf{F}}$.
3. From the correspondences and $\hat{\mathbf{F}}$ estimate the 3D positions of the real-world points $\hat{\mathbf{w}}_i$ using the triangulation method.
4. Given this 3D points project them back to both image planes using the estimate of the camera projection matrices $\hat{\mathbf{x}}_i = \mathbf{P}\hat{\mathbf{w}}_i, \hat{\mathbf{x}}'_i = \mathbf{P'}\hat{\mathbf{w}}_i$ (that were based on $\hat{\mathbf{F}}$).
5. The difference in the real points and the backprojected points is what we want to minimize by varying the camera matrices $\mathbf{P}$ and $\mathbf{P'}$ and the coordinates of the 3D points
6. Minimize the geometric distance from equation 4.29 over $\hat{\mathbf{F}}$ and $\hat{\mathbf{w}}_i$, $i = 1, ..., n$. The cost is minimized using the Levenberg-Marquardt algorithm over 3n + 12 variables: $3n$ for the n 3D points $\hat{\mathbf{w}}_i$, and 12 for the camera matrix.

Chapter 5

HOMOGRAPHY

In order for the eight-point algorithm to give a unique solution (up to a scalar factor) for the camera motion, it is crucial that the feature points in 3D be in general position. When the points happen to form certain degenerate configurations, the solution might no longer be unique. When all the feature points happen to lie on certain 2D surfaces, this is called critical surfaces. Many of these critical surfaces occur rarely in practice and their importance is limited. However, 2D planes, which happen to be a special case of critical surfaces, are ubiquitous in man-made environments and in aerial imaging [6]. Therefore, if one applies the eight-point algorithm to images of points all lying on the same 2D plane, the algorithm will fail to provide a unique solution. On the other hand, in many applications, a scene can indeed be approximately planar (e.g., the landing pad for a helicopter) or piecewise planar (e.g., the corridors inside a building).

5.1 Planar homography

Let us consider two images of a point p on a 2D plane P in 3D space. For simplicity, we will assume throughout the section that the optical center of the camera never passes through the plane, as illustrated in Figure 5.1.

Now suppose that two images $(\mathbf{x}_1, \mathbf{x}_2)$ are given for a point $p \in P$ with respect to two camera frames. Let the coordinate transformation between the two frames be

$$\mathbf{X}_2 = \mathbf{R}\mathbf{X}_1 + \mathbf{T}, \tag{5.1}$$

where $\mathbf{X}_1, \mathbf{X}_2$ are the coordinates of p relative to camera frames 1 and 2, respectively. As we have already seen, the two images $\mathbf{x}_1, \mathbf{x}_2$ of p satisfy the epipolar constraint

$$\mathbf{x}_2^T \mathbf{E} \mathbf{x}_1 = \mathbf{x}_2^T \mathbf{T}_\times \mathbf{R} \mathbf{x}_1 = 0. \tag{5.2}$$

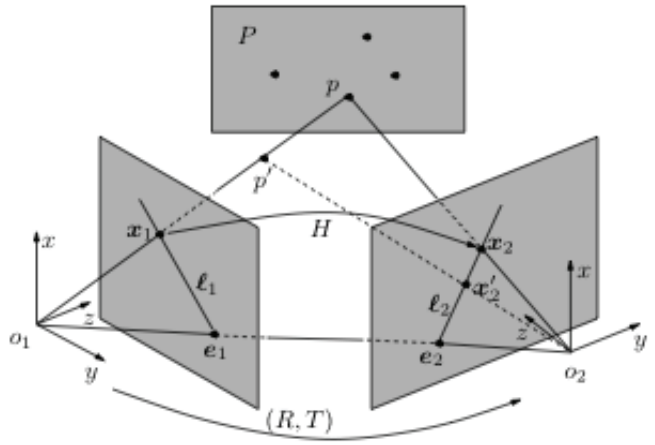


Figure 5.1: Two images $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^3$ of a 3-D point p on a plane P. They are related by a homography $\mathbf{H}$ that is induced by the plane. The figure was copied from [6].

However, for points on the same plane P, their images will share an extra constraint that makes the epipolar constraint alone no longer sufficient.

Let $\mathbf{N} = [n_1, n_2, n_2]^T \in \mathbb{S}^2$ be the unit normal vector of the plane P with respect to the first camera frame, and let $d > 0$ denote the distance from the plane P to the optical center of the first camera. Then we have

$$\mathbf{N}^T \mathbf{X}_1 = n_1 X + n_2 Y + n_3 Z = d \Leftrightarrow \frac{1}{d} \mathbf{N}^T \mathbf{X}_1 = 1, \quad \forall \mathbf{X}_1 \in P. \tag{5.3}$$

Substituting equation (5.3) into equation (5.1) gives

$$\mathbf{X}_2 = \mathbf{R}\mathbf{X}_1 + \mathbf{T} = \mathbf{R}\mathbf{X}_1 + \mathbf{T}\frac{1}{d}\mathbf{N}^T \mathbf{X}_1 = \left(\mathbf{R} + \frac{1}{d}\mathbf{T}\mathbf{N}^T\right)\mathbf{X}_1. \tag{5.4}$$

We call the matrix

$$\mathbf{H} = \mathbf{R} + \frac{1}{d}\mathbf{T}\mathbf{N}^T \quad \in \mathbb{R}^{3\times 3} \tag{5.5}$$

the $(planar)$ *homography matrix*, since it denoted a linear transformation from $\mathbf{X}_1 \in \mathbb{R}^3$ to $\mathbf{X}_1 \in \mathbb{R}^3$ as

$$\mathbf{X}_2 = \mathbf{H}\mathbf{X}_1. \tag{5.6}$$

Note that the matrix $\mathbf{H}$ depends on the motion parameters $\{\mathbf{R}, \mathbf{T}\}$ as well as the structure parameters $\{\mathbf{N}, d\}$ of the plane P. Due to the inherent scale ambiguity in the term $\frac{1}{d}T$ in equation

(5.5), one can at most expect to recover from $\mathbf{H}$ the ratio of the translation $\mathbf{T}$ scaled by the distance d. From

$$\lambda_1 \mathbf{x}_1 = \mathbf{X}_1, \quad \lambda_2 \mathbf{x}_2 = \mathbf{X}_2, \quad \mathbf{X}_2 = \mathbf{H}\mathbf{X}_1, \tag{5.7}$$

we have

$$\lambda_2 \mathbf{x}_2 = \mathbf{H}\lambda_1 \mathbf{x}_1 \Leftrightarrow \mathbf{x}_2 \sim \mathbf{H}\mathbf{x}_1, \tag{5.8}$$

where we recall that $\sim$ indicates equality up to scalar factor. Often, the equation

$$\mathbf{x}_2 \sim \mathbf{H}\mathbf{x}_1 \tag{5.9}$$

itself is referred to as a *(planar) homography* mapping induces by a plane P. Despite the scale ambiguity $\mathbf{H}$ introduces a special map between points in the first image and those in the second in the following sense:

1. For any point $\mathbf{x}_1$ in the first image, that is the image of some point, say p on the plane P, its corresponding second image $\mathbf{x}_2$ is uniquely determined as $\mathbf{x}_2 \sim \mathbf{H}\mathbf{x}_1$ since for any other point, say $\mathbf{x'}_2$, on the same epipolar line $\ell_2 \sim \mathbf{E}\mathbf{x}_1$, the ray $o_2\mathbf{x'}_2$ will intersect the ray $o_1\mathbf{x}_1$ at a point p' out of the plane.

2. On the other hand, if $\mathbf{x}_1$ is the image of some point, say p', not on the plane P, then $\mathbf{x}_2 \sim \mathbf{H}\mathbf{x}_1$ is only a point that is on the same epipolar line $\ell_2 \sim \mathbf{E}\mathbf{x}_1$ as its actual corresponding image $\mathbf{x'}_2$. That is, $\ell_2^T \mathbf{x}_2 = \ell_2^T \mathbf{x'}_2 = 0$.

In addition to the fact that the homography matrix $\mathbf{H}$ encodes information about the camera motion and the scene structure, knowing it directly facilitates establishing correspondence between points in the first and the second images. As we will see soon, $\mathbf{H}$ can be computed in general from a small number of corresponding image pairs. Once $\mathbf{H}$ is known, correspondence between images of other points on the same plane can then be fully established, since the corresponding location $\mathbf{x}_2$ for an image point $\mathbf{x}_1$ is simply $\mathbf{H}\mathbf{x}_1$.

5.2 Estimating the planar homography matrix

We begin with a simple linear algorithm for determining $\mathbf{H}$ given a set of four 2D to 2D point correspondences, $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$. The transformation is given by the equation $\mathbf{x'}_i = \mathbf{H}\mathbf{x}_i$. Note that this is an equation involving homogeneous vectors; thus the 3-vectors $\mathbf{x'}_i$ and $\mathbf{H}\mathbf{x}_i$, are not equal, they have the same direction but may differ in magnitude by a non-zero scale factor. The equation may be expressed in terms of the vector cross product as $\mathbf{x'}_i \times \mathbf{H}\mathbf{x}_i = \mathbf{0}$. This form will enable a simple linear solution for $\mathbf{H}$ to be derived.

If the j-th row of the matrix $\mathbf{H}$ is denoted by $\mathbf{h}^{jT}$, then we have

$$\mathbf{H}\mathbf{x}_i = \begin{pmatrix} \mathbf{h}^{1T}\mathbf{x}_i \\ \mathbf{h}^{2T}\mathbf{x}_i \\ \mathbf{h}^{3T}\mathbf{x}_i \end{pmatrix} \tag{5.10}$$

Writing $\mathbf{x'}_i = (x'_i, y'_i, w'_i)^T$, the cross product can be given explicitly as

$$\mathbf{x'}_i \times \mathbf{H}\mathbf{x}_i = \begin{pmatrix} y'_i\mathbf{h}^{3T}\mathbf{x}_i - w'_i\mathbf{h}^{2T}\mathbf{x}_i \\ w'_i\mathbf{h}^{1T}\mathbf{x}_i - x'_i\mathbf{h}^{3T}\mathbf{x}_i \\ x'_i\mathbf{h}^{2T}\mathbf{x}_i - y'_i\mathbf{h}^{1T}\mathbf{x}_i \end{pmatrix}. \tag{5.11}$$

Since $\mathbf{h}^{jT}\mathbf{x}_i = \mathbf{x}_i^T\mathbf{h}^j$ for $j = 1\ldots, 3$, this gives a set of three equations in the entries of $\mathbf{H}$, which may be written in the form

$$\begin{bmatrix} \mathbf{0}^T & -w'_i\mathbf{x}_i^T & y'_i\mathbf{x}_i^T \\ w'_i\mathbf{x}_i^T & \mathbf{0}^T & -x'_i\mathbf{x}_i^T \\ -y'_i\mathbf{x}_i^T & x'_i\mathbf{x}_i^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix} = \mathbf{0}. \tag{5.12}$$

These equations have the form $\mathbf{A}_i\mathbf{h} = \mathbf{0}$, where $\mathbf{A}_i$, is a 3×9 matrix, and $\mathbf{h}$ is a 9-vector made up of the entries of the matrix $\mathbf{H}$,

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix}, \quad \mathbf{H} \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \tag{5.13}$$

Although equation 5.12 constrains three equations, only two of them are linearly independent. Thus each point correspondence gives two equations in the entries of $\mathbf{H}$. The set of equations can be written as

$$\begin{bmatrix} \mathbf{0}^T & -w'_i\mathbf{x}_i^T & y'_i\mathbf{x}_i^T \\ w'_i\mathbf{x}_i^T & \mathbf{0}^T & -x'_i\mathbf{x}_i^T \end{bmatrix} \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix} = \mathbf{0}. \tag{5.14}$$

Given a set of four point correspondences from the plane, a set of equations $\mathbf{A}\mathbf{h} = \mathbf{0}$ is obtained, where $\mathbf{A}$ is the matrix obtained by stacking the rows of $\mathbf{A}_i$ contributed from each correspondence and $\mathbf{h}$ is the vector of unknown entries of $\mathbf{H}$.

In practise, the extracted image points do not satisfy the relation $\mathbf{x'} = \mathbf{H}\mathbf{x}$ because of noise in the extracted image points. Let us assume that $\mathbf{x'}_i$ is corrupted by Gaussian noise with the mean $\mathbf{0}$ and covariance matrix $\mathbf{\Sigma}_{\mathbf{x'}}$. Then, the maximum likehood estimation of $\mathbf{H}$ is obtained by minimizing the following functional

$$\mathbf{J} = \sum (\mathbf{x'}_i - \hat{\mathbf{x}}'_i)^T \mathbf{\Sigma}_{\mathbf{x'}}^{-\mathbf{1}} (\mathbf{x'}_i - \hat{\mathbf{x}}'_i), \tag{5.15}$$

where

$$\hat{\mathbf{x}}'_i = \frac{\mathbf{1}}{\mathbf{h^{3T}}\mathbf{x}_i} \begin{bmatrix} \mathbf{h^{1T}}\mathbf{x}_i \\ \mathbf{h^{2T}}\mathbf{x}_i \end{bmatrix} \tag{5.16}$$

with $\mathbf{h}^i$ being the ith row of $\mathbf{H}$. In practice, assume $\mathbf{\Sigma}_{\mathbf{x}'} = \sigma^2\mathbf{I}$ for all i. The above problem becomes a nonlinear least-squares estimation problem, i.e., finding $\mathbf{H}$ such that $\|\mathbf{x}'_i - \hat{\mathbf{x}}'_i\|^2$ is minimum. Then

$$\begin{bmatrix} \mathbf{x}^T & 0 & -u\mathbf{x}^T \\ 0 & \mathbf{x}^T & -v\mathbf{x}^T \end{bmatrix} \mathbf{h} = 0. \tag{5.17}$$

Given n points, they can be written in matrix equation as $\mathbf{Ah} = \mathbf{0}$, where $\mathbf{A}$, is a $2n \times 9$ matrix. We seek a non-zero solution $\mathbf{h}$ that minimizes a suitable cost function subject to the constraint $\|\mathbf{h}\| = 1$. This is identical to the problem of finding the minimum of the quotient $\|\mathbf{Ah}\|/\|\mathbf{h}\|$. The solution is the (unit) eigenvector of $\mathbf{A^TA}$ with the least eigenvalue. Equivalently, the solution is the right singular vector associated with the smaller singular value of $\mathbf{A}$ [4].

Algorithm 5 Direct Linear Transformation (DLT) in points

Goal: Given $n \geq 4$ 2D to 2D point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ determine the 2D homography matrix $\mathbf{H}$ such that $\mathbf{x}'_i = \mathbf{Hx}_i$.

1. For each correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ compute $\mathbf{A}_i$. Usually only two rows needed.
2. Assemble $n2 \times 9$ matrices $\mathbf{A}_i$ into a single $2n \times 9$ matrix $\mathbf{A}$.
3. Obtain SVD of $\mathbf{A}$ as $\mathbf{USV^T}$ with $\mathbf{S}$ diagonal with positive diagonal entries, arranged in descending order down the diagonal, then $\mathbf{h}$ is the last column of $\mathbf{V}$.
4. Determine $\mathbf{H}$ from $\mathbf{h}$.

In $\mathbf{A}$, some elements are constant 1, some are in pixels, some are in world coordinates, and some are multiplication of both. This makes $\mathbf{A}$ poorly conditioned numerically. Much better is suggested as follows.

1. Transform the image coordinates according to the transformations $\tilde{\mathbf{x}}_i = \mathbf{Tx}_i$ and $\tilde{\mathbf{x}}'_i = \mathbf{T}'\mathbf{x}'_i$.
2. Find the transformation $\tilde{\mathbf{H}}$ from the correspondences $\tilde{\mathbf{x}}_i \leftrightarrow \tilde{\mathbf{x}}'_i$.
3. Set $\mathbf{H} = \mathbf{T}'\tilde{\mathbf{H}}\mathbf{T}$.

Hartley et al. [4] shows that data normalization gives dramatically better results and hence should be considered as an essential step in the algorithm 5. One of the commonly used transformation is to translate the points so that their centroid is at the origin and the points are scaled such that the average distance from the origin is equal to $\sqrt{2}$.

So far it was assumed that the only source of error in the set of correspondences, $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ is in the measurement of positions. In many practical situations this assumption is not valid because correspondences are computed automatically (SIFT and SURF detectors) and are often mismatched. The mismatched points can be considered as outliers to a Gaussian distribution that explains the error in measurements. These outliers can severely disturb the estimated homography and should be identified. The goal then is to determine a set of inliers from the

presented correspondences so that the homography can be estimated in an optimal manner from these inliers. This is robust estimation sice the estimation is robust or tolerant to outliers, i.e., measurements following a different, and possibly unmodelled, error distribution.

The RANSAC algorithm can be applied to the putative correspondences to estimate the homography and the (inlier) correspondences which are consistent with this estimate. The sample size is four, since four correspondences determine a homography. The number of samples is set adaptively as the proportion of outliers is determined from each consensus state.

Algorithm 6 RANSAC for Homography.

Goal: Compute the homography between the two images given a set of candidate matches

1. Select four points from the set of candidate matches, and compute homography.
2. Select all the pairs which agree with the homography. A pair $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, is considered to agree with a homography $\mathbf{H}$, if $d(\mathbf{H}\mathbf{x}_i, \mathbf{x}'_i) < t$, for some threshold t and $d(.)$ is the Euclidean distance between two pairs.
3. Repeat steps 1 and 2 until a sufficient number of pairs are consistent with the computed homography.
4. Recompute the homography using all consistent correspondences.

There are some important issues in robust estimation using the above procedure. The distance threshold t should be chosen, such that the point is an inlier with a probability a. This calculation requires known probability distribution for the distance of an inlier from the model. In practice, the distance threshold t is chosen empirically so that the probability a that the point is an inlier is high, such as, 0.95. Secondly, trying every possible sample may be prohibitively expensive. Instead a large number of samples is used so that at least one of the random samples of 4 points is free from outliers with a high probability, such as, 0.99. Another rule of thumb employed is to terminate the iterations if the size of the consensus set T is similar to the number of inliers believed to be in the data set. Given the assumed proportion of outliers, we can use $T = (1 - t)n$ for n data points.

5.3 Camera Calibration

From Chapter 3, we learned that the essential parameters of a camera under the pinhole model are its focal length and the size of the image plane (which defines the field of the view of the camera). Also, since we are dealing with the digital images, the number of pixels on the image plane is another important characteristic of a camera. Finally, in order to be able to compute the position of an image's scene point in pixel coordinates, we need one additional piece of information. Considering the line coming from the focal point that is orthogonal to the image plane, we need to know at which pixel position this line pierces the image plane. This point is called the *principal point*. It could be logical to assume that this principal point is at the center of the image plane, but in practice, this one might be off by few pixels depending at which precision the camera has been manufactured.

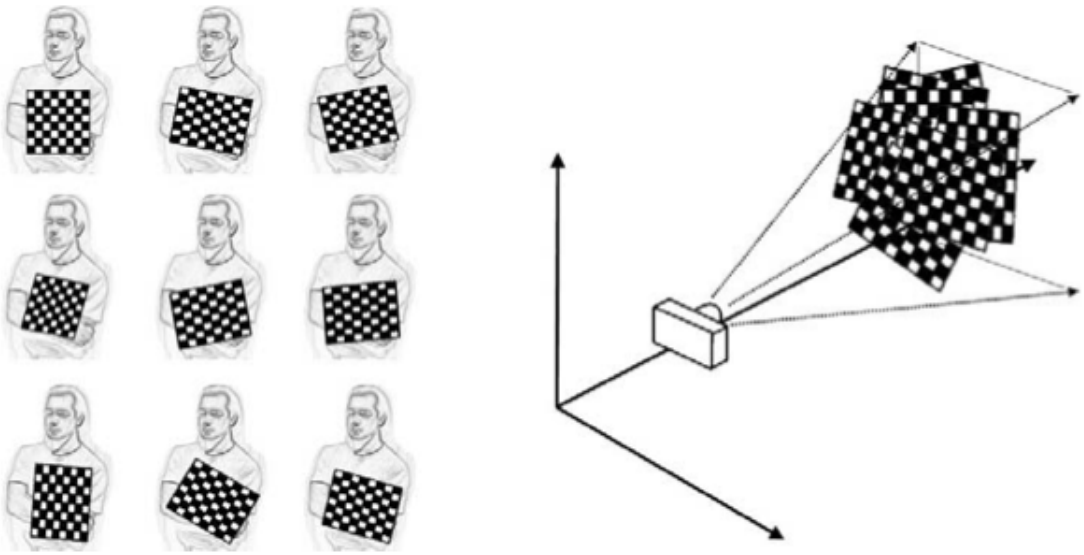


Figure 5.2: Images of a chessboard being held at various orientations (left) provide enough information to completely solve for the locations of those images in global coordinates (relative to the camera) and the camera intrinsics. The figure was copied from [7].

To calibrate a camera, the idea is to show to this camera a set of scene points for which their 3D position is known. You must then determine where on the image these points project. Obviously, for accurate results, we need to observe several of these points. One way to achieve this would be to take one picture of a scene with many known 3D points. A more convenient way would be to take several images from different viewpoints of a set of some 3D points. This approach is simpler but requires computing the position of each camera view, in addition to the computation of the internal camera parameters which fortunately is feasible.

Although there are many ways to solve for the camera parameters, OpenCV choose one that works well on planar objects. The algorithm OpenCV uses to solve for the focal lengths an the offsets is based on Zhang's method [30]. For each view of the chessboard, we collect a homography $\mathbf{H}$ as described previously. We will write $\mathbf{H}$ out as column vectors, $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3]$, where each $\mathbf{h}$ is a 3-by-1 vector. Then, in view of the preceding homography discussion, we can set $\mathbf{H}$ equal to the intrinsic matrix $\mathbf{\Lambda}$ multiplied by a combination of the first two rotation matrix columns, $\mathbf{r}_1$ and $\mathbf{r}_2$, and the translation vector $\mathbf{t}$; after including the scale factor s, this yield:

$$\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3] = s\mathbf{\Lambda}[\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}]. \tag{5.18}$$

Reading off these equations, we have:

$$\begin{aligned} \mathbf{h}_1 &= s\mathbf{\Lambda}\mathbf{r}_1 \;\; or \;\; \mathbf{r}_1 = \lambda\mathbf{\Lambda}^{-1}\mathbf{h}_1 \\ \mathbf{h}_2 &= s\mathbf{\Lambda}\mathbf{r}_2 \;\; or \;\; \mathbf{r}_2 = \lambda\mathbf{\Lambda}^{-1}\mathbf{h}_2 \\ \mathbf{h}_3 &= s\mathbf{\Lambda}\mathbf{t} \;\; or \;\; \mathbf{t} = \lambda\mathbf{\Lambda}^{-1}\mathbf{h}_3 \end{aligned} \tag{5.19}$$

where $\lambda = 1/s$

The rotation vectors are orthogonal to each other by construction, and since the scale is extracted it follows that $\mathbf{r}_1$ and $\mathbf{r}_2$ are orthogonal. Orthogonal implies two things: the rotation vector's dot product is 0, and the vector's magnitudes are equal. Starting with the dot product, we have:

$$\mathbf{r}_1^T \mathbf{r}_2 = 0. \tag{5.20}$$

For any vectors a and b we have $(ab)^T = b^T s^T$, so we can substitute for $\mathbf{r}_1$ and $\mathbf{r}_2$ to derive our first constraint:

$$\mathbf{h}_1^T \mathbf{\Lambda}^{-T} \mathbf{\Lambda}^{-1} \mathbf{h}_2 = 0, \tag{5.21}$$

where $\mathbf{A}^{-T}$ is shorthand for $(\mathbf{A}^{-1})^T$. We also know that the magnitudes at the rotation vectors are equal:

$$\|\mathbf{r}_1\| = \|\mathbf{r}_2\| \quad or \quad \mathbf{r}_1^T \mathbf{r}_1 = \mathbf{r}_2^T \mathbf{r}_2. \tag{5.22}$$

Substituting for $\mathbf{r}_1$ and $\mathbf{r}_2$ yields our second constraint:

$$\mathbf{h}_1^T \mathbf{\Lambda}^{-T} \mathbf{\Lambda}^{-1} \mathbf{h}_1 = \mathbf{h}_2^T \mathbf{\Lambda}^{-T} \mathbf{\Lambda}^{-1} \mathbf{h}_2 \tag{5.23}$$

To make things easier, we set $\mathbf{B} = \mathbf{\Lambda}^{-T} \mathbf{\Lambda}^{-1}$. Writing this out, we have:

$$\mathbf{B} = \mathbf{\Lambda}^{-T} \mathbf{\Lambda}^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}. \tag{5.24}$$

It so happens that this matrix $\mathbf{B}$ has a general closed-form solution:

$$\mathbf{B} = \begin{bmatrix} \frac{1}{f_x^2} & 0 & \frac{-c_x}{f_x^2} \\ 0 & \frac{1}{f_y^2} & \frac{-c_y}{f_y^2} \\ \frac{-c_x}{f_x^2} & \frac{-c_y}{f_y^2} & \frac{-c_x^2}{f_x^2} + \frac{-c_y^2}{f_y^2} + 1 \end{bmatrix}. \tag{5.25}$$

Using the $\mathbf{B}$-matrix, both constraints have the general form $\mathbf{h_i}^T \mathbf{B} \mathbf{h_j}$ in them. Let's multiply this out to see what the components are. Because $\mathbf{B}$ is symmetric, it can be written as one six-dimensional vector dot product. Arranging the necessary elements of $\mathbf{B}$ into the new vector $\mathbf{b}$, we have:

$$\mathbf{h}_i^T \mathbf{B} \mathbf{h}_j = \mathbf{v}_{ij}^T \mathbf{b} = \begin{bmatrix} \mathbf{h}_{i1}\mathbf{h}_{j1} \\ \mathbf{h}_{i1}\mathbf{h}_{j2} + \mathbf{h}_{i2}\mathbf{h}_{j1} \\ \mathbf{h}_{i2}\mathbf{h}_{j2} \\ \mathbf{h}_{i3}\mathbf{h}_{j1} + \mathbf{h}_{i1}\mathbf{h}_{j3} \\ \mathbf{h}_{i3}\mathbf{h}_{j2} + \mathbf{h}_{i2}\mathbf{h}_{j1} \\ \mathbf{h}_{i3}\mathbf{h}_{j3} \end{bmatrix}^T \begin{bmatrix} B_{11} \\ B_{12} \\ B_{22} \\ B_{13} \\ B_{23} \\ B_{33} \end{bmatrix}^T \tag{5.26}$$

Using this definition for $\mathbf{v_{ij}}^T$ our two constraints may now be written as:

$$\begin{bmatrix} \mathbf{v_{12}}^T \\ (\mathbf{v_{11}} - \mathbf{v_{22}})^T \end{bmatrix} \mathbf{b} = 0. \tag{5.27}$$

If we collect K images of chessboards together, then we can stack K of these equations together:

$$\mathbf{V}\mathbf{b} = 0 \tag{5.28}$$

where $\mathbf{V}$ is a $2K$-by-6 matrix. If $K \geq 2$ then this equation can be solved for our $\mathbf{b} = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T$. The camera intrinsics are then pulled directly out of our closed-form solution for the $\mathbf{B}$-matrix:

$$\begin{aligned} f_x &= \sqrt{\lambda / B_{11}} \\ f_y &= \sqrt{\lambda B_{11} / (B_{11}B_{22} - B_{12}^2)} \\ c_x &= -B_{13} f_x^2 / \lambda \\ c_y &= (B_{12}B_{13} - B_{11}B_{23}) / (B_{11}B_{22} - B_{12}^2) \end{aligned} \tag{5.29}$$

where $\lambda = B_{33} - (B_{13}^2 + c_y(B_{12}B_{13} - B_{11}B_{23}))/B_{11}$.

The extrinsics (rotation and translation) are then computed from the equations we read off the homography condition:

$$\begin{aligned} \mathbf{r}_1 &= \lambda \mathbf{\Lambda}^{-1} \mathbf{h_1} \\ \mathbf{r}_2 &= \lambda \mathbf{\Lambda}^{-1} \mathbf{h_2} \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} &= \lambda \mathbf{\Lambda}^{-1} \mathbf{h_3} \end{aligned} . \tag{5.30}$$

Here the scaling parameter is determined from the orthonormality condition $\lambda = 1/\|\mathbf{\Lambda}^{-1}\mathbf{h}_1\|$. Some care is required because, when we solve using real data and put the $\mathbf{r}$-vectors together = $(\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3])$, we will not end up with an exact rotation matrix for which $\mathbf{R}^T\mathbf{R} = \mathbf{R}\mathbf{R}^T = \mathbf{I}$ holds. This is an instance of the *orthogonal Procrustes problem* (see $Appendix$ [3]).

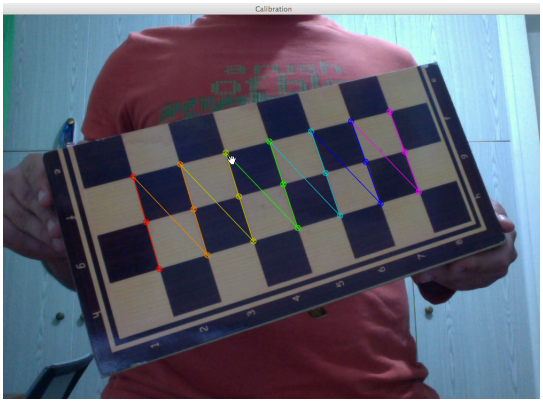


Figure 5.3: Chessboard corners.

OpenCV proposes to use a chessboard pattern to generate the set of 3D scene points required for calibration. This pattern creates points at the corners of each square, and since this pattern is flat, we can freely assume that the board is located at $Z = 0$ with the X and Y axes well aligned with the grid. In this case, the calibration process simply consists of showing the chessboard pattern to the camera from different viewpoints. In figure 5.2 a number of 10 to 20 chessboard images are sufficient, but these must be taken from different viewpoints at different depths. The nice thing is that OpenCV has a *cvFindChessboardCorners* function that automatically detects the corners of this chessboard pattern. You simply provide an image and the size of the chessboard used (number of vertical and horizontal inner corner points) (figure 5.3). The function will return the position of these chessboard corners on the image.

In our calibration example, the reference frame was placed on the chessboard. Therefore, there is a rigid transformation (rotation and translation) that must be computed for each view. These are in the output parameter list of the *cvCalibrateCamera2* function. The rotation and translation components are often called the extrinsic parameters of the calibration and they are different for each view. The intrinsic parameters remain constant for a given camera/lens system. The intrinsic parameters of our test camera obtained from a calibration based on 20 chessboard images are $\phi_x = 9.89290588e + 02$, $\phi_y = 9.89290588e + 02$, $\delta_x = 6.34555359e + 02$, $\delta_y = 3.76046478e + 02$. These results are obtained by *cvCalibrateCamera2* through an optimization process aimed at finding the intrinsic and extrinsic parameters that will minimize the difference between the predicted image point position, as computed from the projection of the 3D scene points, and the actual image point position, as observed on the image. The sums of this difference for all points specified during the calibration is called the *re-projection error*. It is used to quantify how closely an estimate of a 3D point $\hat{\mathbf{x}}$ recreates the point's true projection $\mathbf{x}$.

5.4 Decomposing the planar homography matrix

After we have recovered $\mathbf{H}$ of the form $\mathbf{H} = (\mathbf{R} + \frac{1}{d}\mathbf{T}\mathbf{N}^T)$, we now study how to decompose such a matrix into its motion and structure parameters, namely $\{\mathbf{R}, \frac{\mathbf{T}}{d}, \mathbf{N}\}$ (figure 5.4). First notice

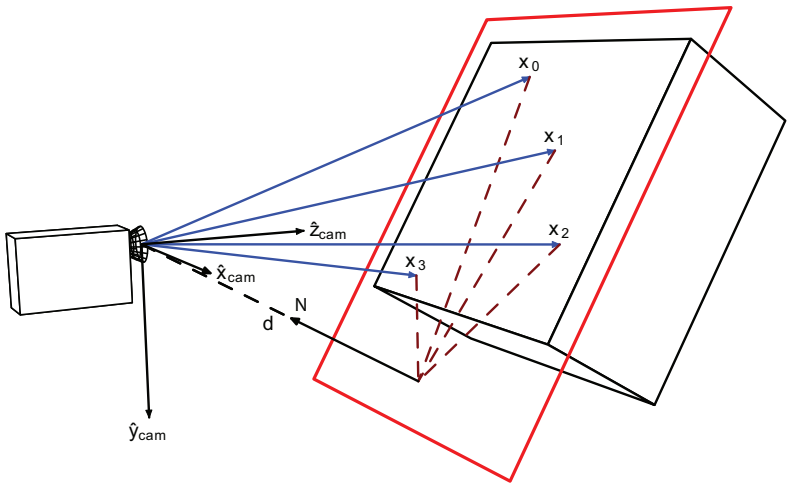


Figure 5.4: The second camera is looking at the plane at distance d. The figure was reproduced from Wikipedia.

that $\mathbf{H}$ preserves the length of any vector orthogonal to $\mathbf{N}$, i.e. if $\mathbf{N} \perp a$, we have $\|\mathbf{N}a\|^2 = \|\mathbf{R}a\|^2 = \|a\|^2$. Also, if we know the plane spanned by the vectors that are orthogonal to $\mathbf{N}$, we then know $\mathbf{N}$ itself. Let us first recover the vector $\mathbf{N}$ based on this knowledge.

The symmetric matrix $\mathbf{H}^T\mathbf{H}$ will have three eigenvalues $\sigma_1^2 \geq \sigma_2^2 \geq \sigma_3^2 \geq 0$. Since $\mathbf{H}^T\mathbf{H}$ is symmetric, it can be diagonalized by an orthogonal matrix $\mathbf{V}$ such that

$$\mathbf{H}^T\mathbf{H} = \mathbf{V}\mathbf{S}\mathbf{V}^T, \tag{5.31}$$

when $\mathbf{S} = \mathbf{diag}\{\sigma_1^2, \sigma_2^2, \sigma_3^2\}$. If $[\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}]$ are the three column vectors of $\mathbf{V}$, we have

$$\mathbf{H}^T\mathbf{H}\mathbf{v}_1 = \sigma_1^2\mathbf{v}_1, \quad \mathbf{H}^T\mathbf{H}\mathbf{v}_2 = \mathbf{v}_2, \quad \mathbf{H}^T\mathbf{H}\mathbf{v}_3 = \sigma_3^2\mathbf{v}_3. \tag{5.32}$$

Hence $\mathbf{v}_2$ is orthogonal to both $\mathbf{N}$ and $\mathbf{T}$, and its length is preserved under the map $\mathbf{H}$. Also, it is easy to check that the length of two other unit-lenght vectors defined as

$$\mathbf{u}_1 = \frac{\sqrt{1-\sigma_3^2}\mathbf{v}_1 + \sqrt{\sigma_1^2 - 1}\mathbf{v}_3}{\sqrt{\sigma_1^2 - \sigma_3^2}}, \quad \mathbf{u}_2 = \frac{\sqrt{1-\sigma_3^2}\mathbf{v}_1 - \sqrt{\sigma_1^2 - 1}\mathbf{v}_3}{\sqrt{\sigma_1^2 - \sigma_3^2}} \tag{5.33}$$

is also preserved under the map $\mathbf{H}$. Furthermore, it is easy to verify that $\mathbf{H}$ preserves the length of any vectors inside each of the two subspaces

$$S_1 = span\{\mathbf{v}_2, \mathbf{u}_1\}, \quad S_2 = span\{\mathbf{v}_2, \mathbf{u}_1\}. \tag{5.34}$$

Since $\mathbf{v}_2$ is orthogonal to $\mathbf{u}_1$ and $\mathbf{u}_2$, $\mathbf{v}_2 \times \mathbf{u}_1$ is a unit normal vector to S_1, and $\mathbf{v}_2 \times \mathbf{u}_2$ a unit

normal vector to S_2. Then $\{\mathbf{v}_2, \mathbf{u}_1, \mathbf{v}_2 \times \mathbf{u}_1\}$ and $\{\mathbf{v}_2, \mathbf{u}_2, \mathbf{v}_2 \times \mathbf{u}_2\}$ form two sets of orthogonal bases for $\mathbb{R}^3$. Notice that we have

$$\mathbf{R}\mathbf{v}_2 = \mathbf{H}\mathbf{v}_2, \quad \mathbf{R}\mathbf{u}_i = \mathbf{H}\mathbf{u}_i, \quad \mathbf{R}(\mathbf{v}_2 \times \mathbf{u}_i) = \mathbf{H}\mathbf{v}_2 \times \mathbf{H}\mathbf{u}_i \tag{5.35}$$

if $\mathbf{N}$ is the normal to the subspace $S_i, i = 1, 2$, as show show in Figure 5.5.

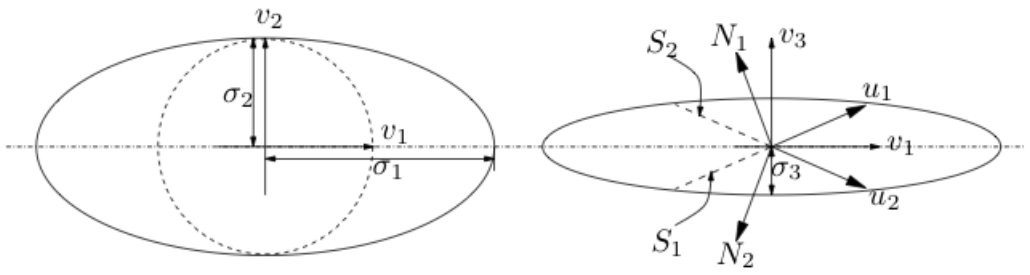


Figure 5.5: In terms of singular vectors $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ and singular values $(\sigma_1, \sigma_2, \sigma_3)$ of the matrix $\mathbf{H}$, there are two candidate subspaces S_1 and S_2 on which the vectors' length is preserved by the homography matrix $\mathbf{H}$.

Define the matrices each of the two subspaces

$$\mathbf{U}_1 = [\mathbf{v}_2, \mathbf{u}_1, \mathbf{v}_2 \times \mathbf{u}_1], \quad \mathbf{W}_1 = [\mathbf{H}\mathbf{v}_2, \mathbf{H}\mathbf{u}_1, \mathbf{H}\mathbf{v}_2 \times \mathbf{H}\mathbf{u}_1], \tag{5.36}$$

$$\mathbf{U}_2 = [\mathbf{v}_2, \mathbf{u}_2, \mathbf{v}_2 \times \mathbf{u}_2], \quad \mathbf{W}_2 = [\mathbf{H}\mathbf{v}_2, \mathbf{H}\mathbf{u}_2, \mathbf{H}\mathbf{v}_2 \times \mathbf{H}\mathbf{u}_2]. \tag{5.37}$$

We then have

$$\mathbf{R}\mathbf{U}_1 = \mathbf{W}_1, \quad \mathbf{R}\mathbf{U}_1 = \mathbf{W}_1. \tag{5.38}$$

This suggests that subspaces S_1 and S_2 may give rise to a solution to the decomposition. By taking into account the extra sign ambiguity in the term $\frac{1}{d}\mathbf{T}\mathbf{N}^T$, we then obtain four solutions for decomposing $\mathbf{H} = (\mathbf{R} + \frac{1}{d}\mathbf{T}\mathbf{N}^T)$ to $\{\mathbf{R}, \frac{\mathbf{T}}{d}, \mathbf{N}\}$. They are given in Table 5.1

Solution 1	$\mathbf{R}_1 = \mathbf{W}_1\mathbf{U}_1^T$ $\mathbf{N}_1 = \mathbf{v}_2 \times \mathbf{u}_1$ $\frac{1}{d}\mathbf{T}_1 = (\mathbf{H} - \mathbf{R}_1)\mathbf{N}_1$	Solution 3	$\mathbf{R}_3 = \mathbf{R}_1$ $\mathbf{N}_3 = -\mathbf{N}_1$ $\frac{1}{d}\mathbf{T}_3 = -\frac{1}{d}\mathbf{T}_1$
Solution 2	$\mathbf{R}_2 = \mathbf{W}_2\mathbf{U}_2^T$ $\mathbf{N}_2 = \mathbf{v}_2 \times \mathbf{u}_2$ $\frac{1}{d}\mathbf{T}_2 = (\mathbf{H} - \mathbf{R}_2)\mathbf{N}_2$	Solution 4	$\mathbf{R}_4 = \mathbf{R}_2$ $\mathbf{N}_4 = -\mathbf{N}_2$ $\frac{1}{d}\mathbf{T}_4 = -\frac{1}{d}\mathbf{T}_2$

Table 5.1: Four solutions for the planar homography decomposition, only two of which satisfy the positive depth constraint.

In order to reduce the number of physically possible solutions, we may impose the positive depth constraint since the camera can see only points that are in front of it, we must have $\mathbf{N}^T e_3 =$

$n_3 > 0$. Suppose that solution 1 is the true one; this constraint will then eliminate solution 3 as being physically impossible. Thus, one of solutions 2 or 4 will be eliminated.

Algorithm 7 The four-point algorithm for planar scene.

Goal: For a given set of image pairs $(\mathbf{x}_{1i}, \mathbf{x}_{2i})$, $i = 1, 2, \ldots, n (n \geq 4)$, of points on a plane $\mathbf{N}^T\mathbf{X} = d$, this algorithm finds $\{\mathbf{R}, \frac{\mathbf{T}}{d}, \mathbf{N}\}$ that solves

$$\mathbf{x}_{2i}^T \times (\mathbf{R} + \tfrac{1}{d}\mathbf{T}\mathbf{N}^T)\mathbf{x}_{1i} = 0, \qquad i = 1, 2, \ldots, n.$$

1. Compute a first approximation of the homography matrix using the DLT and RANSAC algorithm.
2. Normalization of the homography matrix. Compute the eigenvalues $\{\sigma_1, \sigma_2, \sigma_3\}$ of the matrix $\mathbf{H}^T\mathbf{H}$ and normalize it as $\mathbf{H} = \mathbf{H}/\sigma_2$.
3. Decomposition of the homography matrix. Compute the singular value decomposition of
$$\mathbf{H}^T\mathbf{H} = \mathbf{V}\mathbf{S}\mathbf{V}^T$$
and compute the four solutions for a decomposition $\{\mathbf{R}, \frac{\mathbf{T}}{d}, \mathbf{N}\}$ as in Table 5.1.
4. Select the two physically possible ones by imposing the positive depth constraint $\mathbf{N}^T e_3 > 0$.

Similarly with the decomposition of the Essential Matrix in the previous chapter, we have 2-fold ambiguity for the decomposition of Homography. Fortunately, we can resolve this ambiguity using the corresponding pair of points from the two images and the Triangulation algorithm which is developed in the previous Chapter.

5.5 Decomposing the Rotation matrix

From decomposition of $\mathbf{H}$ we recover a rotation matrix $\mathbf{R}$ and a translation matrix $\mathbf{T}$ up to an unknown scaling factor.

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{5.39}$$

Now, we will decompose the given 3×3 rotation matrix to 3 Euler angles [31]

$$\begin{aligned} \theta_x &= atan2(r_{32}, r_{33}) \\ \theta_y &= atan2(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \\ \theta_z &= atan2(r_{21}, r_{11}) \end{aligned} \tag{5.40}$$

Given 3 Euler angles $\theta_x, \theta_y, \theta_z$, the rotation matrix is calculated as follows:

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) \\ 0 & sin(\theta_x) & cos(\theta_x) \end{bmatrix}, \tag{5.41}$$

$$\mathbf{Y} = \begin{bmatrix} cos(\theta_y) & 0 & sin(\theta_y) \\ 0 & 1 & 0 \\ -sin(\theta_y) & 0 & cos(\theta_y) \end{bmatrix}, \tag{5.42}$$

$$\mathbf{Z} = \begin{bmatrix} cos(\theta_z) & -sin(\theta_z) & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.43}$$

Multiplying the three basic rotation matrices we have

$$\mathbf{R} = \mathbf{ZYX}. \tag{5.44}$$

The Euler angles returned when doing a decomposition will be in the following ranges:

$$\begin{aligned} \theta_x &\rightarrow (-\pi, \pi) \\ \theta_y &\rightarrow (-\frac{\pi}{2}, \frac{\pi}{2}) \\ \theta_z &\rightarrow (-\pi, \pi) \end{aligned} \tag{5.45}$$

If we keep angles within these ranges, then we will get the same angles on decomposition. Conversely, if our angles are outside these ranges we will still get the correct rotation matrix, but the decomposed values will be different to our original angles.

Chapter 6

EXPERIMENTAL EVALUATION

Our system consists of five stages, which all must work efficient in order to have an accuracy visual homing process. These stages are illustrated in Figure 6.1. The first stage extracts features from images with the SIFT algorithm, which transforms image data into scale-invariant coordinates relative to local features, and stores them in a database. The second stage, matches each feature from the new image to this previous database and finds candidate matching features based on Euclidean distance of their feature vectors. This is performed with the nearest-neighbor algorithm.

The third stage estimates the homography matrix from four corresponding points of our two planar scenes, which differ by a projective transformation. In the fourth stage, our system calculate the rotation matrix and the translation vector, using the 3D epipolar geometry and the homography. The fifth stage, implements the movement manipulation of our robotic platform and examines if it is located in the target position. If it is not occurred, our robot feeds back a new current image and restart the same visual homing process.

In this chapter, we will examine the robustness of our visual homing process. This will be done by changing some parameters of our homing process and we will evaluate how these parameters affect the visual homing algorithm. Also, we will show how we manipulate the robot movements and how we succeed to move it in the target position.

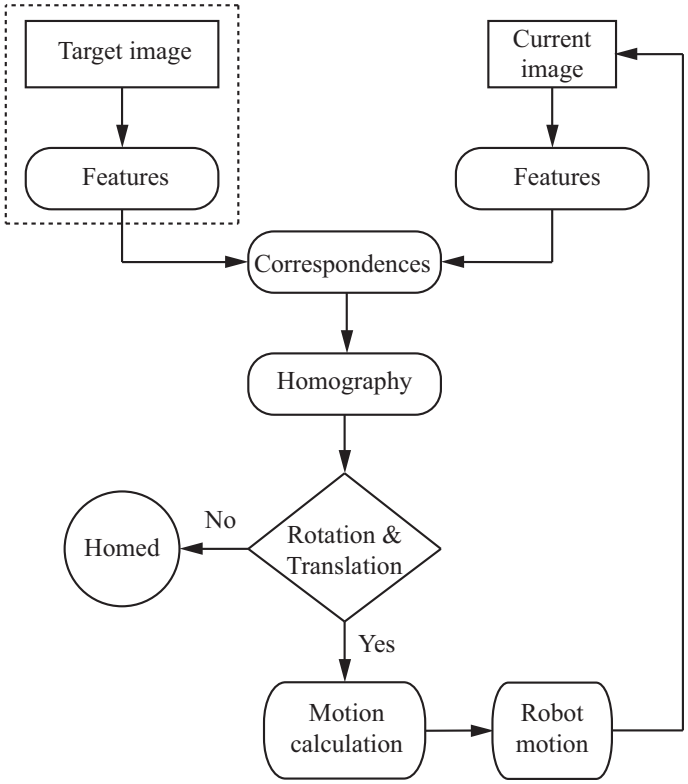


Figure 6.1: Visual homing process.

6.1 Corresponding points

In our visual homing process the rotation must be computed accurately, because even a small rotation error produces high drifts when the robot advances. Also, the rotation is depended from the correct corresponding points. In the application of robot homing we obtain about 25% of wrong matches, but using the appropriate parameters in SIFT descriptors, by eliminating keypoints that have a ratio between the principal curvatures greater than 10, we obtain less than 5% of wrong final matches and even in many cases (depending on the motion and the scenes) we obtain 100% of correct matches.

6.2 Experimental results

We have performed several experiments to evaluate the accuracy with real images computing the motion from homography. We center our attention in the computation of rotation because the translation is recovered at a position only to within a scale factor.

6.2.1 Coordinate system

In order to make the movement manipulations of our robotic platform, we need to know the coordinates system of the motion directions. By decomposing the homography matrix, we take this coordinate systems which is shown in Figure 6.2. As we can see, the y-axis is vertical to our coordinate system and the rotation process is depended by this axis. Also, we can see that the x and z axes, state the distance from the target position.

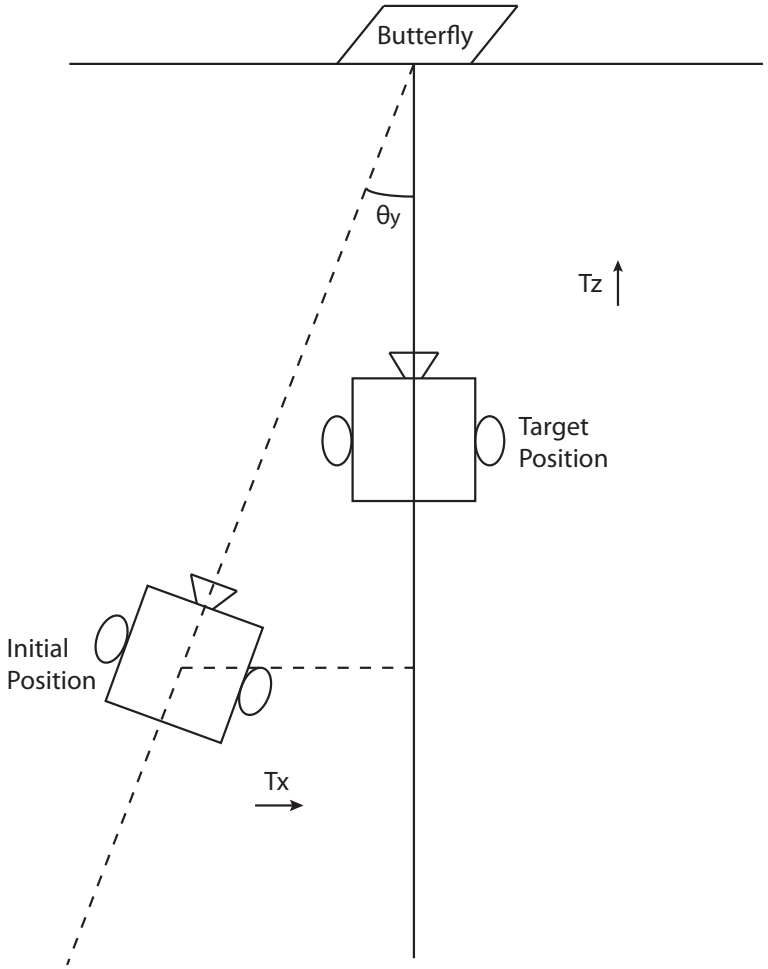


Figure 6.2: Coordinate system of robot's environment.

6.2.2 Homing vectors

We estimate the homography from two planar scenes. First planar scene is shown in Figure 6.3 consider this as a target position. Second planar scenes are Figure 6.4-6.6. We set these, initials position. In each image, we export the rotation matrix and translation vector which are estimated from the decomposition process and we interpret these, in order to move the robotic platform to the target position.

(a)

(b)

Figure 6.3: a) The robot is situated in the target position. b) Image captured by robot in the target position.

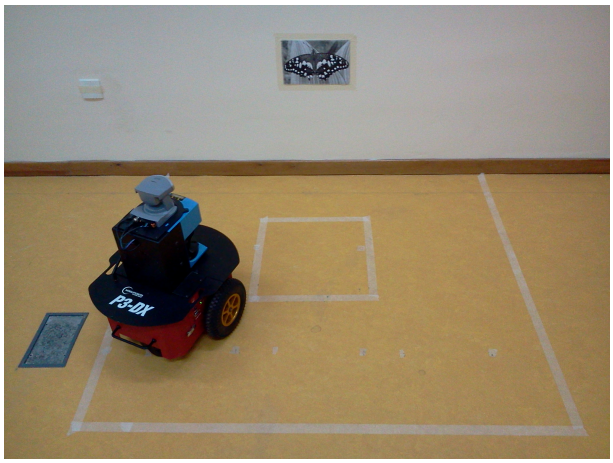
In that position we capture an image from the target position, which is useful for our visual homing process (Figure 6.3). All the other images captured by robot in a arbitrary position, are compared with this image. The advantage with this is that it has all the information that we need in order to find good keypoints and descriptors, by the SIFT and SURF algorithm, and to match them with the descriptors found in the new image, with the nearest-neighbor method. Also, our experimental environment is ideal, because the background is a white wall, and this means that we have no influences from other object. This help us to extract features only from the butterfly image without having outliers by external object, like an indoor environment.

(a)

(b)

Figure 6.4: a) The robot is situated behind our target. b) Image captured by robot in that position

Figure 6.4 has rotation matrix: $\begin{bmatrix} 0.9947 & -0.1011 & 0.0162 \\ 0.1020 & 0.9921 & -0.0729 \\ -0.0087 & 0.0741 & 0.9972 \end{bmatrix}$ and translation vector: $\begin{bmatrix} -0.0985 \\ 0.0451 \\ -0.3550 \end{bmatrix}$. By decomposing the rotation matrix according to the equation (5.40) in Chapter 5, we have $\theta_x = 0.0742$, $\theta_y = 0.0087$ and $\theta_z = 0.1022$. Making the transformation from radians to degrees we take a small rotation. This means that we do not have intense rotation in any axis and the robot is situation behind the target position cause the $t_z < 0$.



(a)

(b)

Figure 6.5: a) The robot is situated left the target position. b) Image captured from left position.

Figure 6.5 has rotation matrix: $\begin{bmatrix} 0.9805 & -0.0986 & -0.1697 \\ 0.0916 & 0.9946 & -0.0491 \\ 0.1736 & 0.0326 & 0.9843 \end{bmatrix}$ and translation vector: $\begin{bmatrix} -0.4908 \\ 0.0043 \\ -0.5676 \end{bmatrix}$. Making the decomposition of rotation matrix, we have $\theta_x = 0.0331$, $\theta_y = -0.1745$ and $\theta_z = 0.091$. This means that we have rotation in y-axis by -20 degrees, the robot is situated behind the target position and has translated left to the target.

(a)

(b)

Figure 6.6: a) The tobot is situated right to the target position. b) Image captured from right position.

Figure 6.6 has rotation matrix: $\begin{bmatrix} 0.9797 & -0.0738 & 0.1864 \\ 0.0837 & 0.9955 & -0.0455 \\ -0.1822 & 0.0601 & 0.9814 \end{bmatrix}$ and translation vector: $\begin{bmatrix} 0.2809 \\ 0.0064 \\ -0.3750 \end{bmatrix}$. Making the decomposition of the rotation matrix, we have $\theta_x = 0.0612$, $\theta_y = 0.1833$ and $\theta_z = 0.0852$. This means that we have rotation in y-axis by 20 degrees, the robot is situation behind the target position and has translated right to the target.

6.2.3 Back-reprojection error

To compute the homography, we have implemented the RANSAC algorithm, which is robust method to consider the existence of outliers. The automatic computing of the homography includes two steps. The first steps is to obtain interest points and determine putative correspondences, while the second one is to estimate the homography and the correspondences which are consistent with this estimate by RANSAC algorithm.

In order to estimate the homography between two planar scenes, we need four corresponding points. When we are not sure for the accuracy of the four corresponding points, we find more pairs with outliers. The RANSAC algorithm can be applied to the putative correspondences to estimate the homography and the (inlier) correspondences which are consistent with this estimate. In Chapter 5, algorithm 5, we set a threshold t which represents the minimum distance where a pair of $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ considered to agree with the homography $\mathbf{H}$.

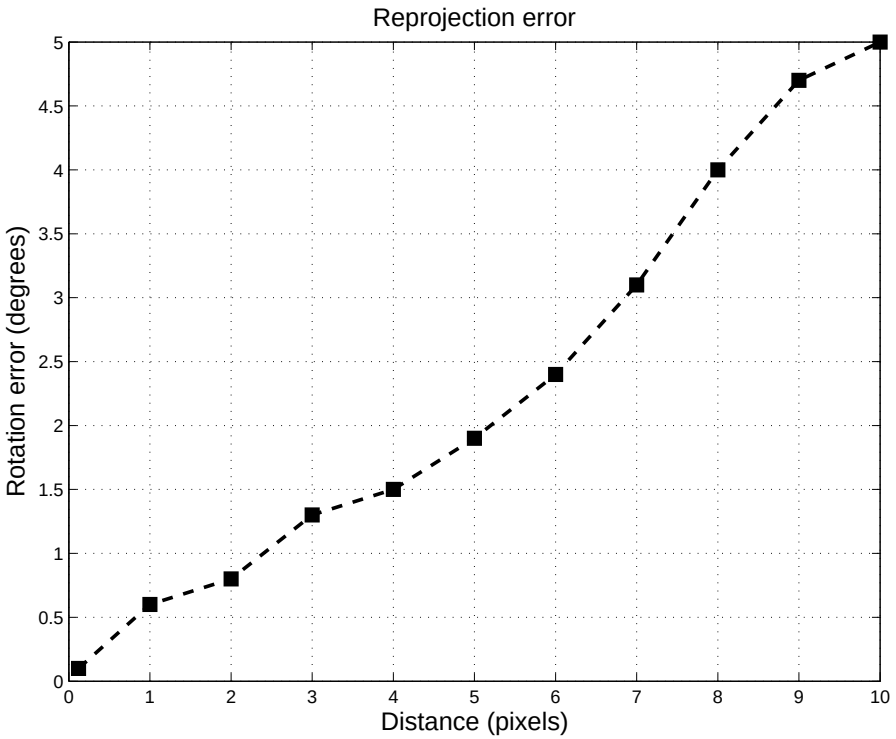


Figure 6.7: Back-reprojection error in rotation angle of the robot using a low resolution camera.

In our experiments we considered this threshold t as maximum allowed reprojection error to treat a point pair as an inlier. That is, if $||dstPoint\mathbf{x}_i - \mathbf{H} \times srcPoint\mathbf{x}'_i|| > ransacReproThresh$ then the point i is considered an outlier. We measure the $srcPoints$ and $dstPoints$ in pixels and set this parameter somewhere in the range of 1 to 10. So, we estimate 10 times the homography with different value of the back-projection error. After that, we decompose each homography to find the rotation matrix and the rotation angle in degrees in y-axis (θ_y). Figure 6.7 represents the rotation error in degrees with the ground truth rotation, as the distance of the pair points, treated as inliers, is increasing in pixels. To test these experiments we take 100 pairs of points from two

standard image position and we check the ground truth error with the estimate rotation from our process.

We can observe that when the threshold is increasing, we have bigger rotation error in our application. This rotation error can produces high drifts when the robot advances. The threshold value 0 is considered to be the ground truth rotation. So we set this threshold to be 1, in order to have the smallest drifts in our navigation. Also, our rotation values are depended from the calibration parameters of our camera. So an important factor for this plot is the accuracy of the calibrated camera. For the first threshold experiments, we use the images taken from Canon VC-C50i Mount PTZ Camera, which adjusted in P3-DX. Their size are 640 x 480 pixels.

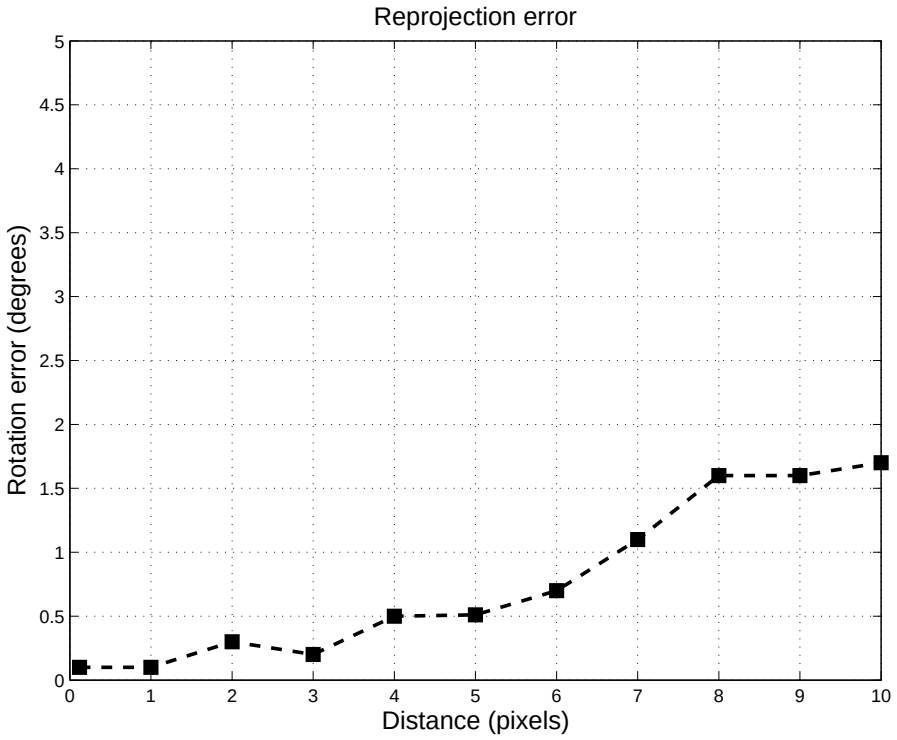


Figure 6.8: Back-reprojection error in rotation angle of the robot using a high resolution camera.

In the second set of experiments we use laptop's camera (iSight) with image resolution 1280 x 960, captured from the same image position with the first experiment. Figure 6.8 shows the improvement, as we have significant reduction in the rotation error estimated by the decomposing process of homography.

6.3 Homography performance with noise

"Image noise" is the digital equivalent of film grain for analogue cameras. Alternatively, one can think of it as analogous to the subtle background hiss you may hear from your audio system at full volume. For digital images, this noise appears as random speckles on an otherwise smooth surface and can significantly degrade image quality. In our application we have to do with digital

images and we set this random noise as a gaussian noise.

We have built a simulator of scenes and motions and we have compared homography decompositions result in different situations in relation to the rotation motion. Image processing and the other not considered errors will be simulated as gaussian random noise added to image coordinate of keypoints found from SIFT algorithm. We can appreciate the influence, on the rotation taken from homography, of image gaussian random noise of zero mean and [0-1] standard deviation in pixels.

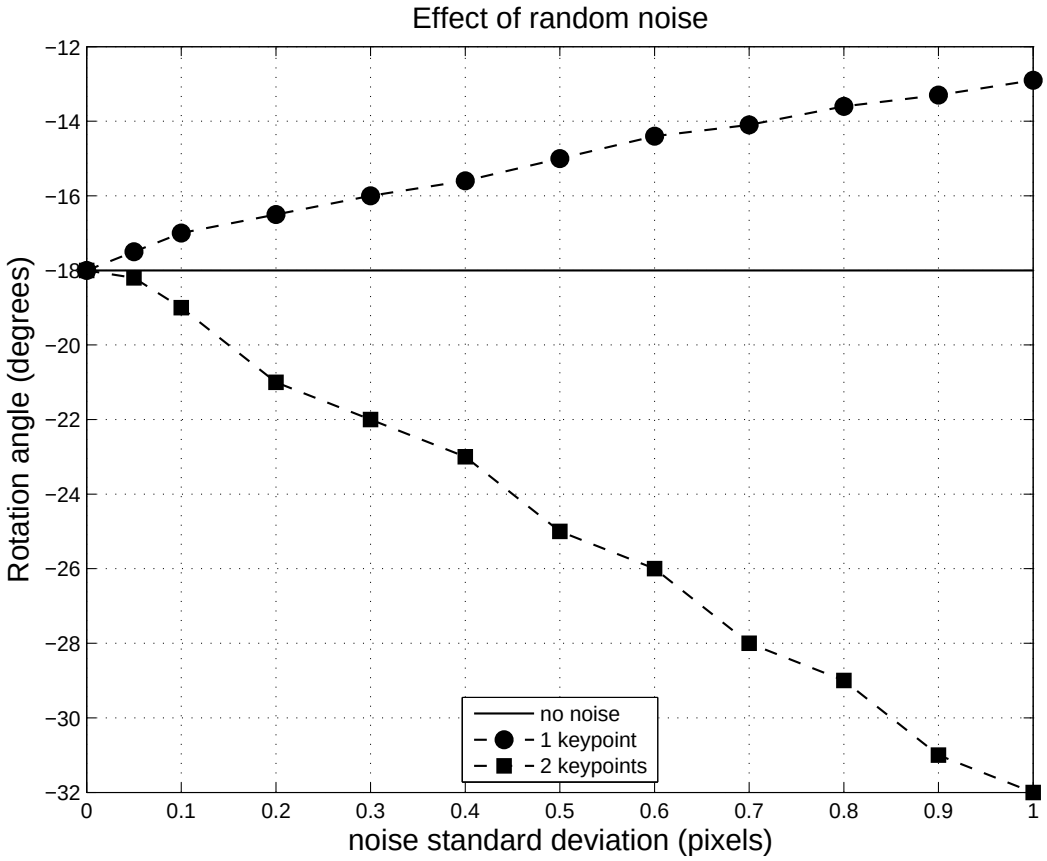


Figure 6.9: Effect of gaussian random noise on computation of the rotation in a low resolution camera.

Figure 6.9 shows the influence of gaussian random noise in a number of keypoints. In order to do that experiments we got a ground truth image take from -18 angles rotation. This means 18 degrees left the target position. We computed the homography matrix from 4 precise corresponding points. Now we try to shift these keypoints from the initial image in order to estimate the different rotation angle in our system. This shift was bounded from 0 to 1 pixel. These results show us that the gaussian noise, affects the estimation of rotation angle and these rotation affects the convergence of our robot in the target. But in our ideal experimental environment these influences are small. In real indoor environments these influences will affect the visual homing navigation and the robotic platform could not reach the target.

In order to overcome these influences, we suggest the robotic platform been occupied with high resolution cameras. Our visual homing process works better with that cameras. We test

these values of gaussian noise in 1280 x 1024 image resolution.

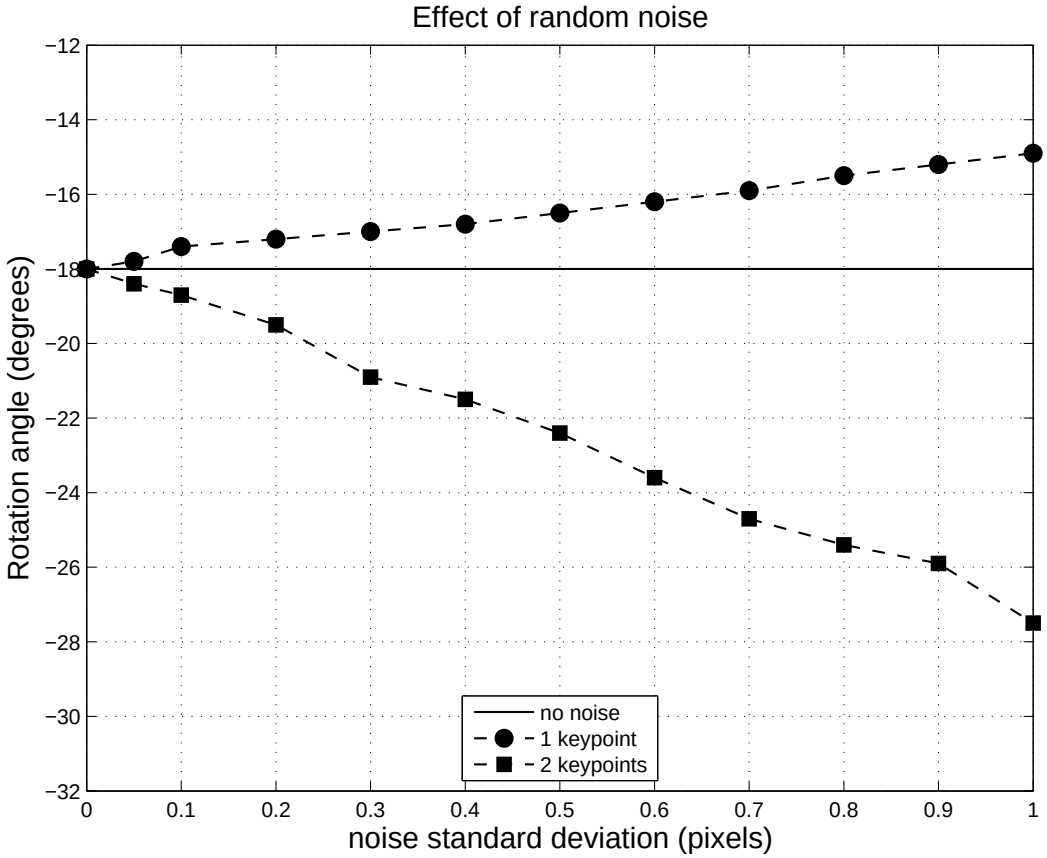


Figure 6.10: Effect of gaussian random noise on computation of the rotation in a high resolution camera.

Figure 6.10 prove the great impact of a high resolution images. Here we have smaller rotation error than the previous resolution images. In a large scale environment like the indoor, this reduction in rotation error is a significant factor for accuracy in robot navigation because it produces low drifts when the robot advances.

6.4 Visual homing experiments

Our experiments were tested in the Robotic Lab in the Department of Computer Science in Ioannina. We use the robotic platform P3-DX with updated operating system Debian 6.0 "squeeze", OpenCV library 2.4 and Canon VC-C50i Mount PTZ Camera.

The only information needed by our system is each current image taken during the navigation and the goal image previously taken at the target position. We set a region near the target position, which our visual homing process could work efficiently. This region play the roll of boundaries, as inside this, the robot can move in the target position without having strange behaviors (lose the target). This happen in indoor environment because the estimated image keypoints are influenced from the low analysis camera of the robot. So, we have the target image

position and the initial image position inside our region. Also, we put the camera's field-of-view at the same object as in the target position. We do that cause our system understand his position relative to this image position. Finally, we start the visual homing process, by estimating the homing vectors which move the robot toward the target position.

The only certain moving command in our robot is the rotation angle relative to the target position. The translation is recovered at another position only to within a scale factor. So, in order to move forward the target position, we make constant movements of 100 mm.

Our experiments are presented in this link:
https://www.youtube.com/watch?v=MJsVNUrlBVo

Chapter 7

CONCLUSION - FUTURE WORK

In the present dissertation, we present a robot navigation procedure using a monocular vision system. The only information needed by the robot P3-DX is each current image taken during navigation and the goal image previously taken at the desired position. The only problem in image-based visual servoing is to deal with the field-of-view constraints of the camera. We overcome this by capturing the same object in our environment. Traditional visual control approaches are based on the epipolar geometry, but this model is ill conditioned for planar scenes. A good alternative is the homography-based approach. It behaves well with planar scenes, which are quite usual in man made environments.

First, in Chapter 2, based on SIFT descriptors which is invariant to image scaling and rotation, we extract features and matching them in the two images. Because the two planar scenes differ by a projective transformation (homography), the first step of the algorithm is to estimate the parameters of the homography. This is done by using the four-point algorithm and RANSAC procedure, presented in Chapter 5. Then, we decompose homography into a rotation matrix and translation vector. Due to a two-fold ambiguity, we use 3D epipolar geometry and the triangulation method to find the correct rotation and an up to scale translation. After that, we import these homing vectors into the robot to estimate his position relative to the target position.

Finally, the performance of our visual homing algorithm was applied in a ideal environment situation and was tested for noise data to simulate environmental conditions, such as the low resolution camera. From our experimental evaluation, we conclude that a high resolution camera make our visual homing process more robust for indoor environment where the accuracy of robot movements are very significant, to reach the target, without lose it. Having look the constraints and the abilities of our visual homing system we can suggest some future work. Occupied our robot with a high resolution camera, and by changing some motion parameters, we can have a better indoor navigation process. Also using a number of different image position in our indoor environment, we can navigate the robot in a certain path. This will be very useful for house navigation. The robot by following different image positions can be in different rooms or ambients, without knowing the localization maps. Furthermore, using the BumbleBee2 stereo

camera, helps the navigation to obtain more information for the relative depth which is strange to received, as our problem have been up to scale.

Bibliography

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, pp. 60(2):91 110, 2004, 2007.

[2] R. Laganière, *OpenCV 2 Computer Vision Application Programming Cookbook*. Packt Pub Limited, 2011. recommended: for OpenCV support.

[3] S. Prince, *Computer vision : models, learning, and inference*. New York: Cambridge University Press, 2012.

[4] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[5] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

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

[7] G. Bradski and A. Kaehler, *Learning OpenCV, [Computer Vision with OpenCV Library ; software that sees]*. O'Reilly Media, 2008. Gary Bradski and Adrian Kaehler.

[8] L. Matthies, M. Maimone, A. Johnson, Y. Cheng, R. Willson, C. Villalpando, S. Goldberg, A. Huertas, A. Stein, and A. Angelova, "Computer vision on mars," *International Journal of Computer Vision*, 2007.

[9] A. E. Johnson, "Precise image-based motion estimation for autonomous small body exploration.," in *Artificial Intelligence Robotics and Automation in Space*, pp. 627–634, 1999.

[10] A. Mourikis, N. Trawny, S. Roumeliotis, A. Johnson, and L. Matthies, "Vision-aided inertial navigation for precise planetary landing: Analysis and experiments," in *Proceedings of Robotics: Science and Systems*, (Atlanta, GA, USA), June 2007.

[11] T. Kanade, O. Amidi, and Q. Ke, "Real-time and 3d vision for autonomous small and micro air vehicles," in *43rd IEEE Conference on Decision and Control (CDC 2004)*, December 2004.

[12] J.-A. Meyer and D. Filliat, "Map-based navigation in mobile robots - ii. a review of map-learning and path-planning strategies," 2003.

[13] K. O. Arras and R. Y. Siegwart, "Feature extraction and scene interpretation for map-based navigation and map building," in *Proc. of SPIE, Mobile Robotics XII*, pp. 42–53, 1997.

[14] G. DeSouza and A. C. Kak., "Vision for mobile robot navigation: A survey.," in *IEEE Trans. on Patt. Analysis and Machine Intelligence*, pp. 24(2):237–267, 2002.

[15] A. A. Argyros, K. E. Bekris, S. C. Orphanoudakis, and L. E. Kavraki, "Robot homing by exploiting panoramic vision," 2005.

[16] D. Bradley, A. Brunton, M. Fiala, and G. Roth, "Image-based navigation in real environments using panoramas," in *In Proceedings of the IEEE International Workshop on Haptic Audio Visual Environments and their Applications*, (Ottawa, Canada), 2007.

[17] J. J. Guerrero, R. Martinez-Cantin, and C. Sagüés, "Visual map less navigation based on homographies," *J. Field Robotics*, vol. 22, no. 10, pp. 569–581, 2005.

[18] C. Sagüés and J. J. Guerrero, "Visual correction for mobile robot homing," *Robotics and Autonomous Systems*, vol. 50, no. 1, pp. 41–49, 2005.

[19] J. Guerrero and C. Sagúes, "Uncalibrated vision based on lines for robot navigation," in *Mechatronics*, p. 11(6):759–777, 2001.

[20] Y. Matsumoto, M. Inaba, and H. Inoue, "Visual navigation using view-sequenced route representation," in *International Conference on Robotics and Automation*, pp. 83–88, 1996.

[21] A. Yohya, Y. Miyazaki, and S. Yuta, "Autonomous navigation of mobile robot based on teaching and playback using trinocular vision," in *International Experts and Consultants*, pp. 389–403, 2001.

[22] R. Basri, E. Rivlin, and I. Shinishonit, "Visual homing: Surfing on the epipoles," in *International Journal of Computer Vision*, pp. 863–869, 1998.

[23] J. S. Pons, W. Hübner, H. Dahmen, and H. A. Mallot, "Vision-based robot homing in dynamic environments," in *International Conference Robotics and Applications*, 2007.

[24] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *In International Conference Computer Vision*, pp. 404–417, 2006.

[25] M. B. Stegmann, "Image warping," in *Informatics and Mathematical Modelling*, 2001.

[26] A. Vardy and R. Möller, "Biologically plausible visual homing methods based on optical flow techniques," *Connection Science, Special Issue: Navigation*, vol. 17, pp. 47–90, 2005.

[27] J. Zeil, M. Hofmann, and J.Chahl, "Catchment areas of panoramic snapshots in outdoor scenes," *Journal of the Optical Society of America*, vol. 20A, no. 3, pp. 450–469, 2003.

[28] D. Ortin and J. M. M. Montiel, "Indoor robot motion based on monocular images," *Robotica*, vol. 19, no. 3, pp. 331–342, 2001.

[29] Q.-T. Luong and O. Faugeras, "The fundamental matrix: theory, algorithms, and stability analysis," *International Journal of Computer Vision*, vol. 17, pp. 43–75, 1995.

[30] Z. Zhang and Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1330–1334, 1998.

[31] T. Herter and K. Lott, "Algorithms for decomposing 3-d orthogonal matrices into primitive rotations," *Computers and Graphics*, vol. 17, no. 5, pp. 517–527, 1993.

Short Vita

Lioulemes Alexandros was born in Gjirokaster of Albania in 1989. He came to Greece in 1998 and stayed tree years in Rizarios boarding school, Monodendri, Zagorochoreia. In 2000 he continued his studies in Ioannina. He was admitted at the Computer Science Department at the University of Ioannina in 2007. He received his BSc degree in 2011 with degree 7.71 "Very Good". That year he became a graduate student at the same institution in which he graduated in July 2013. His main research interest lie in the fields of Computer Vision and Robot Navigation.