Appendices

A Installing the Toolboxes

The Toolboxes are freely available from the book's home page

http://www.petercorke.com/RVC

which also has a lot of additional information related to the book such as web links (all those printed in the book and more), code, figures, exercises and errata.

Downloading and Installing

Two toolboxes support this book: the Robotics Toolbox (RTB) and the Machine Vision Toolbox (MVTB). For the second edition of this book the relevant versions are RTB v10 and MVTB v4.

Toolboxes can be installed from .zip or .mltbx format files, with details below. Once the toolboxes are downloaded you can explore their capability using

```
>> rtbdemo
```

or

```
>> mvtbdemo
```

From .mltbx File

Since MATLAB® R2014b toolboxes can be packaged as, and installed from, files with the extension .mltbx. Download the most recent version of `robot.mltbx` or `vision.mltbx` to your computer. Using MATLAB navigate to the folder where you downloaded the file and double-click it (or right-click then select Install). The Toolbox will be installed within the local MATLAB file structure, and the paths will be appropriately configured for this, and future MATLAB sessions.

From .zip File

Download the most recent version of `robot.zip` or `vision.zip` to your computer. Use your favorite unarchiving tool to unzip the files that you downloaded.

To add the Toolboxes to your MATLAB path execute the command

```
>> addpath RVCDIR ;
>> startup_rvc
```

where `RVCDIR` is the full pathname of the directory where the folder `rvctools` was created when you unzipped the Toolbox files. The script `startup_rvc` adds various subfolders to your path and displays the version of the Toolboxes.

You will need to run the `startup_rvc` script each time you start MATLAB. Alternatively you can run `pathtool` and save the path configuration created by `startup_rvc`.

For installation from zip files, the files for both Toolboxes reside in a top-level directory called `rvctools` and beneath this are a number of subdirectories:

robot	The Robotics Toolbox.
vision	The Machine Vision Toolbox.
common	Utility functions common to the Robotics and Machine Vision Toolboxes.
simulink	Simulink® blocks for robotics and vision, as well as examples.
contrib	Code written by third-parties.

MEX-Files

Some functions in the Toolbox are implemented as MEX-files, that is, they are written in C for computational efficiency but are callable from MATLAB just like any other function. Source code is provided in the `mex` folder along with instructions and scripts to build the MEX-files from inside MATLAB or from the command line. You will require a C-compiler in order to build these files, but prebuilt MEX-files for a limited number of architectures are included.

Contributed Code

A number of useful functions are provided by third-parties and wrappers have been written to make them consistent with other Toolbox functions. If you attempt to access a contributed function that is not installed you will receive an error message.

The contributed code `contrib.zip` can be downloaded, expanded and then added your MATLAB path. If you installed the Toolboxes from .zip files then expand `contrib.zip` inside the folder `RVCDIR`.

Many of these contributed functions are part of active software projects and the downloadable file is a snapshot that has been tested and works as described in this book.

Getting Help

A Google group at http://tiny.cc/rvcforum provides answers to frequently asked questions, and has a user forum for discussing questions, issues and bugs.

License

All the non-third-party code is released under the LGPL license. This means you are free to distribute it in original or modified form provided that you keep the license and authorship information intact.

The third-party code modules are provided under various open-source licenses. The Toolbox compatibility wrappers for these modules are provided under compatible licenses.

MATLAB Versions

The Toolbox software for this book has been developed and tested using MATLAB R2015b and R2016a under Mac OS X (10.11 El Capitan). MATLAB continuously evolves so older versions of MATLAB are increasingly unlikely to work. Please do not report bugs if you are using a MATLAB version older than R2014a.

Octave

GNU Octave (www.octave.org) is an impressive piece of free software that implements a language that is close to, but not the same as, MATLAB. The Toolboxes will not work well with Octave, though with Octave 4 the incompatibilities are greatly reduced. An old version of the arm-robot functions described in Chap. 7–9 have been ported to Octave and this code is distributed in `RVCDIR/robot/octave`.

Appendix

B Linear Algebra Refresher

B.1 Vectors

A rank 1 tensor.

We will only consider real vectors which are an ordered *n-tuple* of real numbers $v_1, v_2, \cdots v_n$ which is usually written as

$$\boldsymbol{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \quad \text{or} \quad \boldsymbol{v} = (v_1, v_2, \cdots v_n)$$

which are a colum- and row-vector respectively. These are equivalent to an $n \times 1$ and a $1 \times n$ matrix respectively, and can be multiplied with a conforming matrix.

The numbers v_1, v_2 etc. are called the scalar components of $\boldsymbol{v}$, and v_i is called the i^{th} component of $\boldsymbol{v}$. For a 3-vector we often write the elements as $\boldsymbol{v} = (v_x, v_y, v_z)$.

The symbol $\mathbb{R}^n$ represents the set of ordered n-tuples of real numbers, each vector is a point in this space, that is $\boldsymbol{v} \in \mathbb{R}^n$. The elements of $\mathbb{R}^2$ can be represented in a plane by a point or a directed line segment. The elements of $\mathbb{R}^3$ can be represented in a volume by a point or a directed line segment.

A vector space is an n-dimensional space whose elements are vectors *plus* the operations of addition and scalar multiplication. The addition of any two elements $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ yields $(a_1 + b_1, a_2 + b_2 \cdots a_n + b_n)$ and $s\boldsymbol{a} = (sa_1, sa_2 \cdots sa_n)$. Both results are element of $\mathbb{R}^n$. The negative of a vector is obtained by negating each element of the vector $-\boldsymbol{a} = (-a_1, -a_2 \cdots -a_n)$.

We can use a vector to represent a point with coordinates $(x_1, x_2, \cdots x_n)$ which is called a coordinate vector. However we need to be careful because the operations of addition and scalar multiplication, while valid for vectors are meaningless for points. We can add a vector to the coordinate vector of a point to obtain the coordinate vector of another point, and we can subtract one coordinate vector from another, and the result is the is the displacement between the points.

The magnitude or length of a vector is a nonnegative scalar given by its p-norm

$$\|v\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p}$$

The Euclidean length of a vector is given by $\|v\|_2$ which is also referred to as the L_2 norm and is generally assumed when p is omitted, for example $\|v\|$. A unit vector is one where $\|v\|_2 = 1$ and is denoted as $\hat{\boldsymbol{v}}$. The L_1 norm is sum of the absolute value of the elements and is also known as the Manhattan distance, it is the distance traveled when confined to moving along the lines in a grid. The L_∞ norm is the maximum element of the vector.

The dot product of two column vectors is a scalar

$$\boldsymbol{a} \cdot \boldsymbol{b} = \boldsymbol{b} \cdot \boldsymbol{a} = \boldsymbol{a}^T \boldsymbol{b} = \boldsymbol{b}^T \boldsymbol{a} = \sum_{i=1}^{n} a_i b_i = \|a\|_2 \|b\|_2 \cos\theta$$

where θ is the angle between the vectors. $\boldsymbol{a} \cdot \boldsymbol{b} = 0$ when the vectors are orthogonal. For 3-vectors the cross product

$$\boldsymbol{a} \times \boldsymbol{b} = -\boldsymbol{b} \times \boldsymbol{a} = \det \begin{pmatrix} \hat{\boldsymbol{x}} & \hat{\boldsymbol{y}} & \hat{\boldsymbol{z}} \\ a_1 & a_2 & a_3 \\ b_1 & b_1 & b_3 \end{pmatrix} = [a]_\times \boldsymbol{b} = \|a\|_2 \|b\|_2 \sin\theta \hat{\boldsymbol{n}}$$

where $\hat{\boldsymbol{x}}$ is a unit-vector parallel to the x-axis etc., $[\cdot]_\times$ is a skew-symmetric matrix as described in the next section, and $\hat{\boldsymbol{n}}$ is a unit-vector normal to the plane containing $\boldsymbol{a}$ and $\boldsymbol{b}$. If the vectors are parallel $\boldsymbol{a} \times \boldsymbol{b} = 0$.

B.2 Matrices

A taxonomy of matrices is shown in Fig. B.1. In this book we are concerned only with real $m \times n$ matrices

Real matrices are a subset of all matrices. For the general case of complex matrices the term Hermitian is the analog of symmetric, and unitary the analog of orthogonal. A^H denotes the Hermitian transpose, the complex conjugate transpose of the complex matrix $\boldsymbol{A}$. Matrices are rank 2 tensors.

$$\boldsymbol{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \\ a_{m,1} & a_{n,2} & \cdots & a_{m,n} \end{pmatrix}, \quad \boldsymbol{A} \in \mathbb{R}^{m \times n}$$

with m rows and n columns. If $n = m$ the matrix is square.

The transpose is

$$\boldsymbol{B} = \boldsymbol{A}^T, \; b_{i,j} = a_{j,i} \quad \forall i, j$$

and it can be shown that

$$(\boldsymbol{AB})^T = \boldsymbol{B}^T \boldsymbol{A}^T, \; (\boldsymbol{ABC})^T = \boldsymbol{C}^T \boldsymbol{B}^T \boldsymbol{A}^T, \; \text{etc.}$$

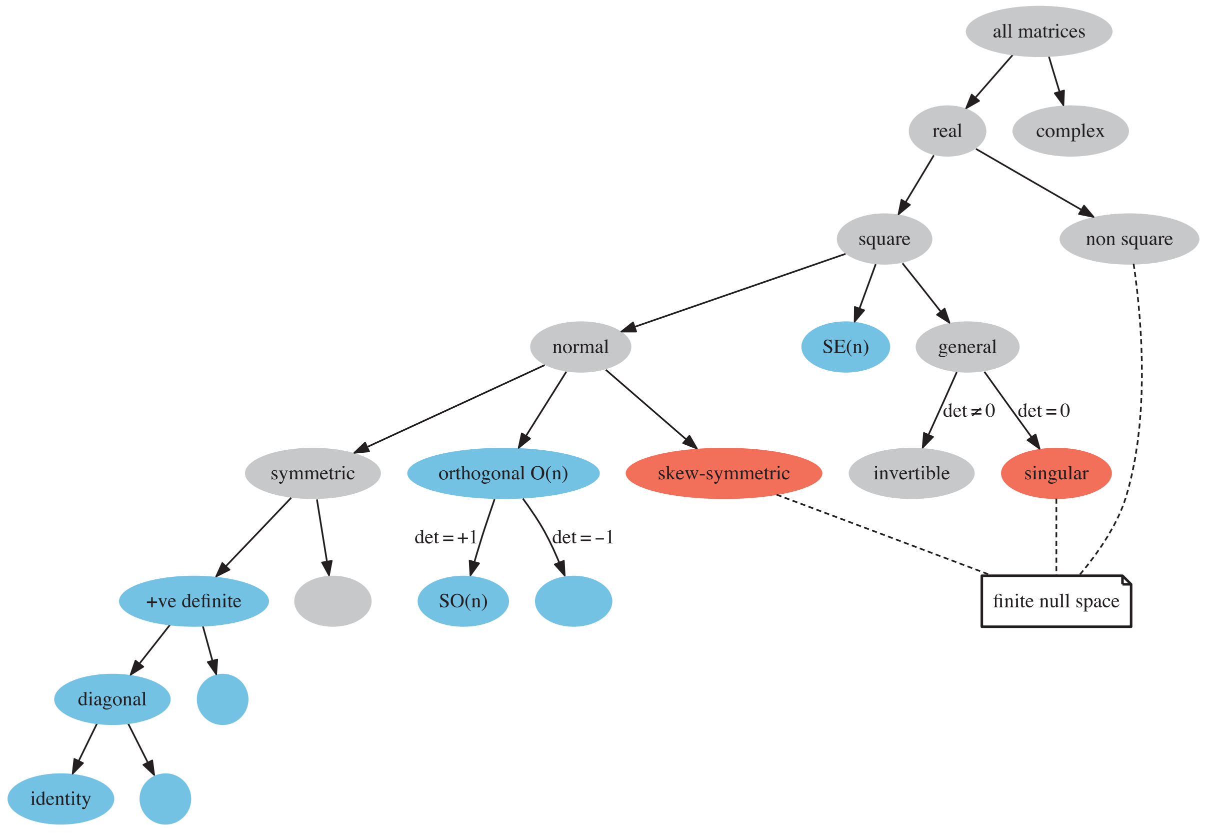


Fig. B.1. Taxonomy of matrices. Classes of matrices that are always singular are shown in *red*, those that are never singular are shown in *blue*

B.2.1 Square Matrices

A square matrix may have an inverse $\boldsymbol{A}^{-1}$ in which case

`Ai = inv(A)`

$$\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}_{n\times n}$$

where

$$\boldsymbol{I}_{n\times n} = \begin{pmatrix} 1 & & 0 \\ & 1 & \\ & & \ddots & \\ 0 & & & 1 \end{pmatrix} \in \mathbb{R}^{n\times n}$$

is the identity matrix, a unit diagonal matrix. The inverse exists provided that the matrix is nonsingular, that is, its determinant $\det(\boldsymbol{A}) \neq 0$. The inverse can be computed from the matrix of cofactors. If $\boldsymbol{A}$ and $\boldsymbol{B}$ are square and nonsingular then

$$(\boldsymbol{A}\boldsymbol{B})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}, \ \ (\boldsymbol{A}\boldsymbol{B}\boldsymbol{C})^{-1} = \boldsymbol{C}^{-1}\boldsymbol{B}^{-1}\boldsymbol{A}^{-1}, \ \text{ etc.}$$

and also

$$\left(\boldsymbol{A}^T\right)^{-1} = \left(\boldsymbol{A}^{-1}\right)^T$$

The inverse can be written as

$$\boldsymbol{A}^{-1} = \frac{1}{\det(\boldsymbol{A})}\operatorname{adj}(\boldsymbol{A})$$

where $\operatorname{adj}(\boldsymbol{A})$ is the transpose of the matrix of cofactors and known as the adjugate or adjoint matrix and sometimes denoted by $\boldsymbol{A}^*$. If $\boldsymbol{B} = \operatorname{adj}(\boldsymbol{A})$ then $\boldsymbol{A} = \operatorname{adj}(\boldsymbol{B})$. If $\boldsymbol{A}$ is nonsingular the adjugate can be computed by

$$\operatorname{adj}(\boldsymbol{A}) = \det(\boldsymbol{A})\boldsymbol{A}^{-1}$$

For a square matrix if

$\boldsymbol{A} = \boldsymbol{A}^T$ .... the matrix is **symmetric**. The inverse of a symmetric matrix is also symmetric. Many matrices that we encounter in robotics are symmetric, for example covariance matrices and manipulator inertia matrices.

`S = skew(v)`

$\boldsymbol{A} = -\boldsymbol{A}^T$ .... the matrix is **skew-symmetric** or **anti-symmetric**. Such a matrix has a zero diagonal, is always singular and has the property that $[a\boldsymbol{v}]_\times = a[\boldsymbol{v}]_\times$, $[\boldsymbol{R}\boldsymbol{v}]_\times = \boldsymbol{R}[\boldsymbol{v}]_\times\boldsymbol{R}^T$ and $\boldsymbol{v}^T[\boldsymbol{v}]_\times = [\boldsymbol{v}]_\times\boldsymbol{v} = 0, \forall\boldsymbol{v}$. For the 3×3 case

$$\boldsymbol{S} = [v]_\times = \begin{pmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{pmatrix} \quad \text{(B.1)}$$

`v = vex(S)`

and the inverse operation is

$$\boldsymbol{v} = \operatorname{vex}(\boldsymbol{S})$$

$\boldsymbol{A}^{-1} = \boldsymbol{A}^T$ .... the matrix is **orthogonal**. The matrix is also known as orthonormal since its column vectors (and row vectors) must be of unit length and orthogonal to each other. The product of two orthogonal matrices of the same size is also an orthogonal matrix. The set of $n \times n$ orthogonal matrices forms a group $\mathbf{O}(n)$, known as the orthogonal group. The determinant of an orthogonal matrix is either $+1$ or -1. The subgroup $\mathbf{SO}(n)$ consisting of orthogonal matrices with determinant $+1$ is called the special orthogonal group. The columns (and rows) are orthogonal vectors, that is, their dot product is zero.

$A^TA = AA^T$ ……the matrix is **normal** and can be diagonalized by an orthogonal matrix U so that U^TAU is a diagonal matrix. All symmetric, skew-symmetric and orthogonal matrices are normal matrices as are matrices of the form $A = B^TB = BB^T$ where B is an arbitrary matrix.

The square matrix $A \in \mathbb{R}^{n\times n}$ can be applied as a linear transformation to a vector $x \in \mathbb{R}^n$

$$x' = Ax$$

which results in another vector, generally with a change in its length and direction. However there are some important special cases. If $A \in \mathbf{SO}(n)$ the transformation is isometric and the vector's *length* is unchanged $\|x'\| = \|x\|$.

In 2-dimensions if x is the set of all points lying on a circle then x' defines points that lie on an ellipse. The MATLAB® builtin demonstration

```
>> eigshow
```

shows this very clearly as you interactively drag the tip of the vector x around the unit circle.

The eigenvectors of a square matrix are those vectors x such that

`[x,e] = eig(A)`

$$Ax = \lambda_i x \tag{B.2}$$

that is, their direction is unchanged when transformed by the matrix. They are simply scaled by λ_i, the corresponding eigenvalue. The matrix has n eigenvalues (the *spectrum* of the matrix) which can be real or complex. For an orthogonal matrix the eigenvalues lie on a unit circle in the complex plane, $|\lambda_i| = 1$, and the eigenvectors are all orthogonal to one another.

The eigenvalues of a real symmetric matrix are all real and we classify the matrix according to the sign of its eigenvalues

▪ $\lambda_i > 0, \forall i$	positive definite
▪ $\lambda_i \geq 0, \forall i$	positive semi-definite
▪ $\lambda_i < 0, \forall i$	negative definite
▪ otherwise	indefinite

The inverse of a positive definite matrix is also positive definite.

The matrices A^TA and AA^T are always symmetric and positive semidefinite. This implies than any symmetric matrix A can be written as

$$A = LL^T$$

where L is the Cholesky decomposition of A.

`L = chol(A)`

The matrix R such that

$$A = RR$$

`R = sqrtm(A)`

is the square root of A or $A^{½}$.

If $\boldsymbol{T}$ is any nonsingular matrix then

$$\boldsymbol{A} = \boldsymbol{T}\boldsymbol{B}\boldsymbol{T}^{-1}$$

is known as a similarity transform and $\boldsymbol{A}$ and $\boldsymbol{B}$ are said to be similar, and it can be shown that the eigenvalues are unchanged by the transformation.

If $\boldsymbol{A}$ is nonsingular then the eigenvectors of $\boldsymbol{A}^{-1}$ are the same as $\boldsymbol{A}$ and the eigenvalues of $\boldsymbol{A}^{-1}$ are the reciprocal of those of $\boldsymbol{A}$. The eigenvalues of $\boldsymbol{A}^T$ are the same as those of $\boldsymbol{A}$ but the eigenvectors are different.

The matrix form of Eq. B.2 is

$$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{\Lambda}$$

where $\boldsymbol{X} \in \mathbb{R}^{n\times n}$ is a matrix of eigenvectors of $\boldsymbol{A}$, arranged column-wise, and $\boldsymbol{\Lambda}$ is a diagonal matrix of corresponding eigenvalues. If $\boldsymbol{X}$ is not singular we can rearrange this as

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^{-1}$$

which is the eigenvalue or spectral decomposition of the matrix. This implies that the matrix can be diagonalized by a similarity transform

$$\boldsymbol{\Lambda} = \boldsymbol{X}^{-1}\boldsymbol{A}\boldsymbol{X}$$

If $\boldsymbol{A}$ is symmetric then $\boldsymbol{X}$ is orthogonal and we can instead write

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{\Lambda}\boldsymbol{X}^T \tag{B.3}$$

`det(A)`

The determinant of a square matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the factor by which the transformation changes changes volumes in an n-dimensional space. For 2-dimensions imagine a shape defined by points $\boldsymbol{x}_i$ with an enclosed area a. The shape formed by the points $\boldsymbol{A}\boldsymbol{x}_i$ would have an enclosed area $a\det(A)$. If $\boldsymbol{A}$ is singular the points $\boldsymbol{A}\boldsymbol{x}_i$ would lie at a single point or along a line and have zero enclosed area. In a similar way for 3-dimensions, the determinant is a scale factor applied to the volume of a set of points mapped through the transformation $\boldsymbol{A}$.

The determinant is equal to the product of the eigenvalues

$$\det(\boldsymbol{A}) = \prod_{i=1}^{n} \lambda_i$$

`trace(A)`

thus a matrix with one or more zero eigenvalues will be singular. A positive definite matrix, $\lambda_i > 0$, therefore has $\det(A) > 0$ and is not singular. The trace of a matrix is the sum of the diagonal elements

$$\operatorname{tr}(\boldsymbol{A}) = \sum_{i=1}^{n} A_{ii}$$

which is also the sum of the eigenvalues

$$\operatorname{tr}(\boldsymbol{A}) = \sum_{i=1}^{n} \lambda_i$$

The columns of $\boldsymbol{A} = (\boldsymbol{c}_1\boldsymbol{c}_2 \cdots \boldsymbol{c}_n)$ can be considered as a set of vectors that define a space – the column space. Similarly, the rows of $\boldsymbol{A}$ can be considered as a set of vectors that define a space – the row space. The column rank of a matrix is the number of linearly independent columns of $\boldsymbol{A}$. Similarly, the row rank is the number of linearly independent rows of $\boldsymbol{A}$. The column rank and the row rank are always equal and are simply called the rank of $\boldsymbol{A}$ and the rank has an upper bound of $\min(m, n)$. The rank is the dimension of the largest nonsingular square submatrix that can be formed from $\boldsymbol{A}$. A square matrix for which $\mathrm{rank}(\boldsymbol{A}) < n$ is said to be rank deficient or not of full rank. The rank shortfall $\min(m, n) - \mathrm{rank}(\boldsymbol{A})$ is the nullity of $\boldsymbol{A}$. In addition $\mathrm{rank}\,(\boldsymbol{AB}) \le \min\,(\mathrm{rank}\,(\boldsymbol{A}), \mathrm{rank}\,(\boldsymbol{B}))$ and $\mathrm{rank}\,(\boldsymbol{A}+\boldsymbol{B}) \le \mathrm{rank}\,(\boldsymbol{A}) + \mathrm{rank}\,(\boldsymbol{B})$. The matrix $\boldsymbol{v}\boldsymbol{v}^T$ has rank 1 for all $\boldsymbol{v} \ne 0$.

`rank(A)`

B.2.2 Nonsquare and Singular Matrices

For a nonsquare matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ we can determine the left generalized inverse or pseudo inverse or Moore-Penrose pseudo inverse

$$\boldsymbol{A}^+\boldsymbol{A} = \boldsymbol{I}_{n\times n}$$

where $\boldsymbol{A}^+ = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$. The right generalized inverse is

$$\boldsymbol{A}\boldsymbol{A}^+ = \boldsymbol{I}_{m\times m}$$

where $\boldsymbol{A}^+ = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1}$.

If the matrix $\boldsymbol{A}$ is not of full rank then it has a finite null space or kernel. A vector $\boldsymbol{x}$ lies in the null space of the matrix if

$$\boldsymbol{A}\boldsymbol{x} = 0$$

More precisely this is the right-null space. A vector lies in the left-null space if

$$\boldsymbol{x}\boldsymbol{A} = 0$$

The left null space is equal to the right null space of $\boldsymbol{A}^T$.

The null space is defined by a set of orthogonal basis vectors whose dimension is the nullity of $\boldsymbol{A}$. Any linear combination of these null-space basis vectors lies in the null space.

`null(A)`

For a nonsquare matrix $\boldsymbol{A} \in \mathbf{R}^{m\times n}$ the analog to Eq. B.2 is

$$\boldsymbol{A}\boldsymbol{v}_i = \sigma_i \boldsymbol{u}_i$$

where $\boldsymbol{u}_i \in \mathbf{R}^m$ and $\boldsymbol{v}_i \in \mathbf{R}^n$ are respectively the right- and left-singular vectors of $\boldsymbol{A}$, and σ_i its singular values. The singular values are nonnegative real numbers that are the square root of the eigenvalues of $\boldsymbol{A}\boldsymbol{A}^T$ and $\boldsymbol{u}_i$ are the corresponding eigenvectors. $\boldsymbol{v}_i$ are the eigenvectors of $\boldsymbol{A}^T\boldsymbol{A}$.

The singular value decomposition or SVD of the matrix $\boldsymbol{A}$ is

`[U,S,Vt] = svd(A)`

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

where $\boldsymbol{U} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ are both orthogonal matrices comprising, as columns, the corresponding singular vectors $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$. $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}$ is a diagonal matrix of the singular values

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & \mathbf{0} & \\ & & \sigma_r & & \\ & & & 0 & \\ & \mathbf{0} & & & \ddots & \\ & & & & & 0 \end{pmatrix}$$

where $r = \mathrm{rank}(\boldsymbol{A})$ is the rank of $\boldsymbol{A}$ and $\sigma_i \geq \sigma_{i+1}$. For the case where $r < n$ the diagonal will have zero elements as shown. Columns of $\boldsymbol{V}^T$ corresponding to the zero columns of $\boldsymbol{\Sigma}$ define the null space of $\boldsymbol{A}$. The condition number of a matrix $\boldsymbol{A}$ is $\max \sigma_i / \min \sigma_i$ and a high value means the matrix is close to singular or "poorly conditioned". `cond(A)`

The matrix quadratic form

$$s = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \tag{B.4}$$

is a scalar. If $\boldsymbol{A}$ is positive definite then $s = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} > 0, \forall \boldsymbol{x} \neq 0$.

For the case that $\boldsymbol{A}$ is diagonal this can be written

$$s = \sum_{i=1}^{n} A_{ii} x_i^2$$

which is a weighted sum of squares. If $\boldsymbol{A}$ is symmetric then

$$s = \sum_{i=1}^{n} A_{ii} x_i^2 + 2 \sum_{i=1}^{n} \sum_{j=i+1}^{n} A_{ij} x_i x_j$$

the result also includes products or correlations between elements of $\boldsymbol{x}$.

The Mahalanobis distance is a weighted distance or norm

$$s = \sqrt{\boldsymbol{v}^T \boldsymbol{P}^{-1} \boldsymbol{v}}$$

where $\boldsymbol{P} \in \mathbb{R}^{n \times n}$ is a covariance matrix which down-weights components of $\boldsymbol{v}$ where uncertainty is high.

C Geometry

Geometric concepts such as points, lines, ellipses and planes are critical to the fields of robotics and robotic vision. We briefly summarize key representations in both Euclidean and projective (homogeneous coordinate) space.

C.1 Euclidean Geometry

C.1.1 Points

A point in n-dimensional space is represented by an n-tuple, an ordered set of n numbers $(x_1, x_2 \cdots x_n)$ which define the coordinates of the point. The tuple can also be interpreted as a vector – a coordinate vector – from the origin to the point.

C.1.2 Lines

C.1.2.1 Lines in 2D

A line is defined by $\ell = (a, b, c)$ such that

$$ax + by + c = 0 \tag{C.1}$$

which is a generalization of the line equation we learned in school $y = mx + c$ but which can easily represent a vertical line by setting $a = 0$. $\boldsymbol{v} = (a, b)$ is a vector parallel to the line, and $\boldsymbol{v} = (-b, a)$ is a vector normal to the line. The line that joins two points is given by the solution to

$$\begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = 0$$

which is found from the right-null space of the left-most term. The intersection point of two lines is

$$\begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = -\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

which has no solution if the lines are parallel – the left-most term is singular.

We can also represent the line in polar form

$$\cos\theta x + \sin\theta y + \rho = 0$$

where θ is the angle from the x-axis to the line and ρ is the normal distance between the line and the origin, as shown in Fig. 13.18.

C.1.2.2 Lines in 3D and Plücker Coordinates

We can define a line by two points, $\mathbf{p}$ and $\mathbf{q}$, as shown in Fig. C.1, which would require a total of six parameters $\ell = (q_x, q_y, q_z, p_x, p_y, p_z)$. However since these points can be arbitrarily chosen there would be an infinite set of parameters that represent the same line making it hard to determine the equivalence of two lines.

There are advantages in representing a line as

$$\ell = (\boldsymbol{\omega} \times \boldsymbol{q}, \boldsymbol{p} - \boldsymbol{q}) = (\boldsymbol{v}, \boldsymbol{\omega}) \in \mathbb{R}^6$$

where $\boldsymbol{\omega}$ is the direction of the line and $\boldsymbol{v}$ is the moment of the line – a vector from the origin to a point on the line and normal to the line. This is a Plücker coordinate vector – a six dimensional quantity subject to two constraints: the coordinates are homogeneous and thus invariant to overall scale factor; and $\boldsymbol{v} \cdot \boldsymbol{\omega} = 0$. Lines therefore have 4 degrees-of-freedom▸ and the Plücker coordinates lie on a 4-dimensional manifold in 6-dimensional space. Lines with $\boldsymbol{\omega} = 0$ lie at infinity and are known as ideal lines.▸

This is not intuitive but consider two parallel planes and an arbitrary 3D line passing through them. The line can be described by the 2-dimensional coordinates of its intersection point on each plane – a total of four coordinates.

Ideal as in imaginery, not as in perfect.

In MATLAB® we will first define two points as column vectors

```
>> P = [2 3 4]';  Q = [3 5 7]';
```

and then create a Plücker line object

```
>> L = Plucker(P, Q)
L =
{ 1  -2  1; -1  -2  -3 }
```

which displays the v and w components. These can be accessed as properties

```
>> L.v'
ans =
     1    -2     1
>> L.w'
ans =
    -1    -2    -3
```

A Plücker line can also be represented as a skew-symmetric matrix

```
>> L.L
ans =
     0     1     2    -1
    -1     0     1    -2
    -2    -1     0    -3
     1     2     3     0
```

which can also be formed by $\tilde{\boldsymbol{p}}\tilde{\boldsymbol{q}}^T - \tilde{\boldsymbol{q}}\tilde{\boldsymbol{p}}^T$.

To plot this line we first define a region of 3D space▸ then plot it in blue

Since lines lines are infinite we need to specify a finite volume in which to draw it.

```
>> axis([-5 5  -5 5  -5 5]);
>> L.plot('b');
```

The line is the set of all points

$$\boldsymbol{p}(\lambda) = \frac{\boldsymbol{v} \times \boldsymbol{\omega}}{\boldsymbol{\omega} \cdot \boldsymbol{\omega}} + \lambda \boldsymbol{\omega}, \quad \lambda \in \mathbb{R}$$

which can be generated parametrically in terms of a scalar parameter

```
>> L.point([0 1 2])
ans =
   -0.5714   -1.5714   -2.5714
   -0.1429   -2.1429   -4.1429
    0.2857   -2.7143   -5.7143
```

where the columns are points on the line corresponding to $\lambda = 0, 1, 2$.

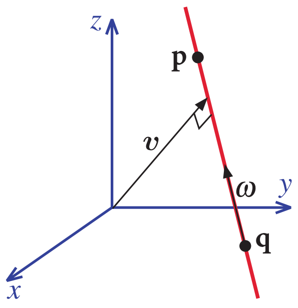


Fig. C.1. Describing a line in 3-dimensions

A point $\boldsymbol{x}$ is closest to the line when

$$\lambda = \frac{(\boldsymbol{x}-\boldsymbol{q})\cdot\boldsymbol{\omega}}{\boldsymbol{\omega}\cdot\boldsymbol{\omega}}$$

For the point (1, 2, 3) the closest point on the line, and its distance, is given by

```
>> [x, d] = L.closest([1 2 3]')
x =
    3.1381
    2.5345
    1.9310
d =
    2.4495
```

The line intersects the plane $\boldsymbol{n}^T\boldsymbol{x} + d = 0$ at the point coordinate

$$\boldsymbol{x} = \frac{\boldsymbol{v}\times\boldsymbol{n} - d\boldsymbol{\omega}}{\boldsymbol{\omega}\cdot\boldsymbol{n}}$$

For the *xy*-plane the line intersects at

```
>> L.plane_intersect([0 0 1 0])'
ans =
    0.6667    0.3333         0
```

Two lines can be identical, coplanar or skewed. Identical lines have linearly dependent Plücker coordinates, that is, $\ell_1 = \lambda\ell_2$. If coplanar they can be parallel or intersecting and if skewed can be intersecting or not. If lines have $\boldsymbol{\omega}_1 \times \boldsymbol{\omega}_2 = 0$ they are parallel otherwise they are skewed.

The minimum distance between two lines is

$$d = \boldsymbol{\omega}_1\cdot\boldsymbol{v}_2 + \boldsymbol{\omega}_2\cdot\boldsymbol{v}_1$$

and is zero if they intersect.

The side operator is a permuted dot product

$$\mathrm{side}\left(\ell^1, \ell^2\right) = \ell^1_1\ell^2_5 + \ell^1_2\ell^2_6 + \ell^1_3\ell^2_4 + \ell^1_4\ell^2_3 + \ell^1_5\ell^2_1 + \ell^1_6\ell^2_2$$

which is zero if the lines intersect or are parallel and is computed by the `side` method.

We can transform a Plücker line by the adjoint of a rigid-body motion.

$$\ell' = \mathrm{Ad}(\xi)\ell$$

Julius Plücker (1801–1868) was a German mathematician and physicist who made contributions to the study of cathode rays and analytical geometry. He was born at Elberfeld and studied at Düsseldorf, Bonn, Heidelberg and Berlin and went to Paris in 1823 where he was influenced by the French geometry movement. In 1825 he returned to the University of Bonn, was made professor of mathematics in 1828, and professor of physics in 1836. In 1858 he proposed that the lines of the spectrum, discovered by his colleague Heinrich Geissler (of Geissler tube fame), were characteristic of the chemical substance which emitted them. In 1865, he returned to geometry and invented what was known as line geometry. He was the recipient of the Copley Medal from the Royal Society in 1866, and is buried in the Alter Friedhof (Old Cemetery) in Bonn.

C.1.3 Planes

A plane is defined by a 4-vector $\pi = (a, b, c, d)$ such that

$$ax + by + cz + d = 0$$

which can be written in point-normal form as

$$\boldsymbol{n}^T(\boldsymbol{x} - \boldsymbol{p}) = 0$$

for a plane containing a point with coordinate $\boldsymbol{p}$ and a normal $\boldsymbol{n}$, or more generally as

$$\boldsymbol{n}^T\boldsymbol{x} + d = 0$$

A plane can be defined by 3 points

$$\begin{pmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{pmatrix} \pi = 0$$

and solved for using the right-null space of the left-most term, or by two nonparallel lines

$$\pi = (\boldsymbol{\omega}_1 \times \boldsymbol{\omega}_2, \boldsymbol{v}_1 \cdot \boldsymbol{\omega}_2)$$

or by a line and a point with coordinate $\boldsymbol{r}$

$$\pi = (\boldsymbol{\omega} \times \boldsymbol{r} - \boldsymbol{v}, \boldsymbol{v} \cdot \boldsymbol{r})$$

A point is defined as the intersection point of three planes

$$\begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = - \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

The Plücker line formed by the intersection of two planes is

$$\ell = (\boldsymbol{n}_1 \times \boldsymbol{n}_2,\ d_2\boldsymbol{n}_1 - d_1\boldsymbol{n}_2)$$

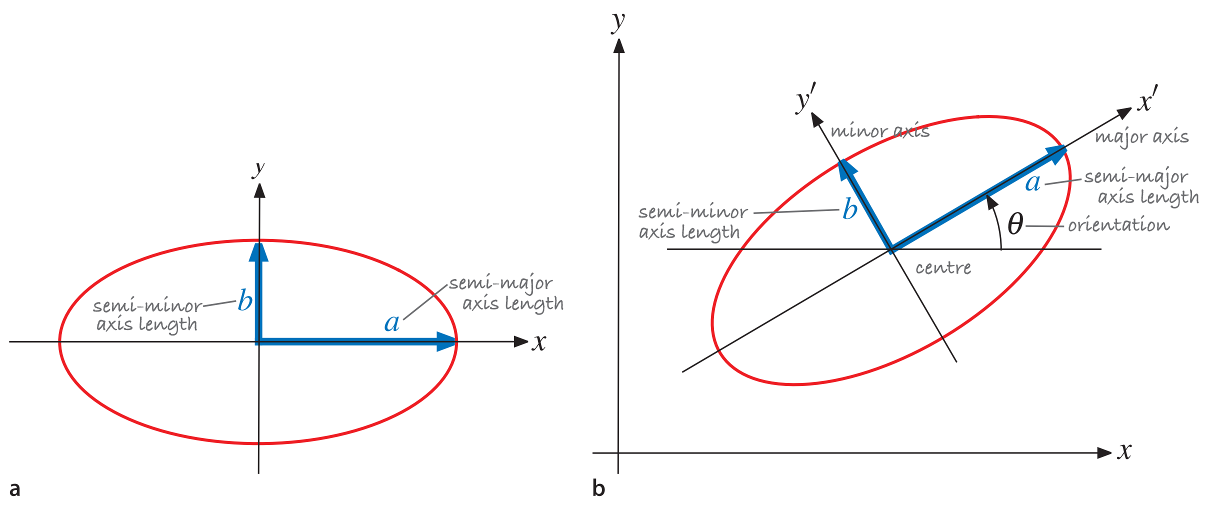


Fig. C.2. Ellipses. **a** Canonical ellipse centered at the origin and aligned with the x- and y-axes; **b** general form of ellipse

C.1.4 Ellipses and Ellipsoids

An ellipse belongs to the family of planar curves known as conics. The simplest form of an ellipse is defined implicitly

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

and is shown in Fig. C.2a. This canonical ellipse is centered at the origin and has its major and minor axes aligned with the x- and y-axes. The radius in the x-direction is a and in the y-direction is b. The longer of the two radii is known as the semi-major axis length and the other is the semi-minor axis length.

We can write the ellipse in matrix quadratic form Eq. B.4 as

$$\begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 1/a^2 & 0 \\ 0 & 1/b^2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 1$$

$$\boldsymbol{x}^T \begin{pmatrix} a^2 & 0 \\ 0 & b^2 \end{pmatrix}^{-1} \boldsymbol{x} = 1 \tag{C.2}$$

$$\boldsymbol{x}^T \boldsymbol{E}^{-1} \boldsymbol{x} = 1 \tag{C.3}$$

In the most general form $\boldsymbol{E}$ is a symmetric matrix

$$\boldsymbol{E} = \begin{pmatrix} A & C \\ C & B \end{pmatrix} \tag{C.4}$$

and its determinant $\det(E) = AB - C^2$ defines the type of conic

$$\det(\boldsymbol{E}) \begin{cases} > 0 & \text{ellipse} \\ = 0 & \text{parabola} \\ < 0 & \text{hyperbola} \end{cases}$$

An ellipse is therefore represented by a positive definite symmetric matrix $\boldsymbol{E}$. Conversely any positive definite symmetric matrix, such as an inertia matrix or covariance matrix, can be represented by an ellipse.

Nonzero values of C change the orientation of the ellipse. The ellipse can be arbitrarily centered at $\boldsymbol{x}_c$ by writing it in the form

$$(\boldsymbol{x} - \boldsymbol{x}_c)^T \boldsymbol{E}^{-1} (\boldsymbol{x} - \boldsymbol{x}_c) = 1$$

which leads to the general ellipse shown in Fig. C.2b.

Since $\boldsymbol{E}$ is symmetric it can be diagonalized by Eq. B.3

$$\boldsymbol{E} = \boldsymbol{X} \boldsymbol{\Lambda} \boldsymbol{X}^T$$

where $\boldsymbol{X}$ is an orthogonal matrix comprising the eigenvectors of $\boldsymbol{E}$. The inverse is

$$\boldsymbol{E}^{-1} = \boldsymbol{X} \boldsymbol{\Lambda}^{-1} \boldsymbol{X}^T$$

so the quadratic form becomes

$$\begin{aligned} \boldsymbol{x}^T \boldsymbol{X} \boldsymbol{\Lambda}^{-1} \boldsymbol{X}^T \boldsymbol{x} &= 1 \\ \left(\boldsymbol{X}^T \boldsymbol{x}\right)^T \boldsymbol{\Lambda}^{-1} \left(\boldsymbol{X}^T \boldsymbol{x}\right) &= 1 \\ \boldsymbol{x}'^T \boldsymbol{\Lambda}^{-1} \boldsymbol{x}' &= 1 \end{aligned}$$

This is similar to Eq. C.3 but with the ellipse defined by the diagonal matrix $\boldsymbol{\Lambda}$ with respect to the rotated coordinated frame $\boldsymbol{x}' = \boldsymbol{X}^T\boldsymbol{x}$. The major and minor ellipse axes are aligned with the eigenvectors of $\boldsymbol{E}$. The squared radii of the ellipse are the eigenvalues of $\boldsymbol{E}$ or the diagonal elements of $\boldsymbol{\Lambda}$.

For the general case of $\boldsymbol{E} \in \mathbb{R}^{n\times n}$ the result is an ellipsoid in n-dimensional space. The Toolbox function `plot_ellipse` will draw an ellipse for the $n = 2$ case and an ellipsoid for the $n = 3$ case.

Alternatively the ellipse can be represented in polynomial form by writing as

$$\left(\boldsymbol{x} - (x_0,\ y_0)\right)^T \begin{pmatrix} a & c \\ c & b \end{pmatrix} \left(\boldsymbol{x} - (x_0,\ y_0)\right) = 1$$

and expanding to obtain

$$e_1x^2 + e_2y^2 + e_3xy + e_4x + e_5y + e_6 = 0$$

where $e_1 = a$, $e_2 = b$, $e_3 = 2c$, $e_4 = -2(ax_0 + cy_0)$, $e_5 = -2(by_0 + cx_0)$ and $e_6 = ax_0^2 + by_0^2 + 2cx_0y_0 - 1$. The ellipse has only five degrees of freedom, its center coordinate and the three unique elements in $\boldsymbol{E}$. For a nondegenerate ellipse where $e_1 \neq 0$ we can rewrite the polynomial in normalized form

$$x^2 + E_1y^2 + E_2xy + E_3x + E_4y + E_5 = 0 \tag{C.5}$$

with five unique parameters.

C.1.4.1 Properties

The area of an ellipse is πab and its eccentricity is

$$\varepsilon = \frac{\sqrt{a^2 - b^2}}{a}$$

The eigenvectors of $\boldsymbol{E}$ define the principal directions of the ellipse and the square root of the eigenvalues are the corresponding radii.

Consider the ellipse

$$\boldsymbol{x}\begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}^{-1} \boldsymbol{x} = 1$$

which is represented in MATLAB by

```
>> E = [2 -1; -1 1];
```

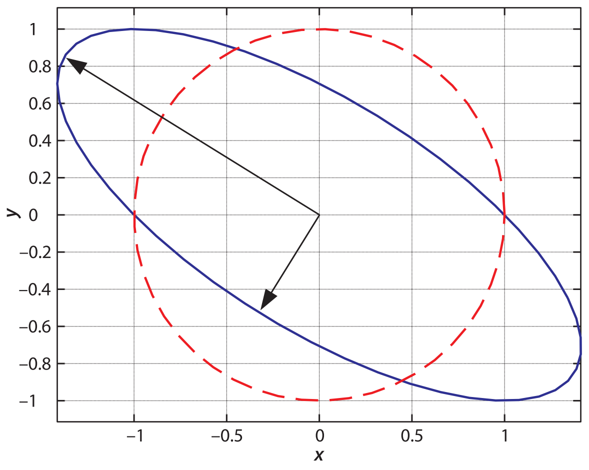


Fig. C.3. Ellipse corresponding to a symmetric 2×2 matrix, and the unit circle shown in *red*. The arrows indicate the major and minor axes of the ellipse

We can plot this by

```
>> plot_ellipse(E)
```

which is shown in Fig. C.3.

The eigenvectors and eigenvalues of $\boldsymbol{E}$ are

```
>> [x,e] = eig(E)
x =
   -0.5257   -0.8507
   -0.8507    0.5257
e =
    0.3820         0
         0    2.6180
```

and the ellipse radii are

```
>> r = sqrt(diag(e))
r =
    0.6180
    1.6180
```

which correspond to b and a respectively. If either radius is equal to zero the ellipse is degenerate and becomes a line. If both radii are zero the ellipse is a point.

The eigenvectors are unit vectors in the minor- and major-axis directions and we will scale them by the radii to yield radius vectors which we can plot

```
>> arrow([0 0]', x(:,1)*r(1));
>> arrow([0 0]', x(:,2)*r(2));
```

The orientation of the ellipse is the angle of the major-axis with respect to the horizontal axis and is

$$\theta = \tan^{-1}\frac{\boldsymbol{x}_y}{\boldsymbol{x}_x}$$

For our example this is

```
>> atan2(x(2,2), x(1,2)) * 180/pi
ans =
  148.2825
```

in units of degrees.

The ellipse area is $\pi r_1 r_2$ and the ellipsoid volume is $\frac{4}{3}\pi r_1 r_2 r_3$ where the radii $r_i = \sqrt{\lambda_i}$ where λ_i are the eigenvalues of $\boldsymbol{E}$. Since $\det(\boldsymbol{E}) = \Pi\lambda_i$ the area or volume is proportional to $\sqrt{\det(\boldsymbol{E})}$.

C.1.4.2 Drawing an Ellipse

In order to draw an ellipse we first define a point coordinate $\boldsymbol{y} = [x, y]^T$ on the unit circle

$$\boldsymbol{y}^T\boldsymbol{y} = 1$$

and rewrite Eq. C.3 as

$$\boldsymbol{x}^T\boldsymbol{E}^{-\frac{1}{2}}\boldsymbol{E}^{-\frac{1}{2}}\boldsymbol{x} = 1$$

where $\boldsymbol{E}^{\frac{1}{2}}$ is the matrix square root (MATLAB function `sqrtm`). Equating these two equations we can write

$$\boldsymbol{x}^T\boldsymbol{E}^{-\frac{1}{2}}\boldsymbol{E}^{-\frac{1}{2}}\boldsymbol{x} = \boldsymbol{y}^T\boldsymbol{y}$$

It is clear that

$$\boldsymbol{y} = \boldsymbol{E}^{-\frac{1}{2}}\boldsymbol{x}$$

which we can rearrange as

$$\boldsymbol{x} = \boldsymbol{E}^{\frac{1}{2}}\boldsymbol{y}$$

which transforms a point on the unit circle to a point on an ellipse. If the ellipse is centered at $\boldsymbol{x}_c$ rather than the origin we can perform a change of coordinates

$$(\boldsymbol{x} - \boldsymbol{x}_c)^T \boldsymbol{E}^{-\frac{1}{2}}\boldsymbol{E}^{-\frac{1}{2}}(\boldsymbol{x} - \boldsymbol{x}_c) = 1$$

from which we write the transformation as

$$\boldsymbol{x} = \boldsymbol{E}^{\frac{1}{2}}\boldsymbol{y} + \boldsymbol{x}_c$$

Continuing the MATLAB example above

```
>> E = [2 -1; -1 1];
```

We define a set of points on the unit circle

```
>> th = linspace(0, 2*pi, 50);
>> y = [cos(th);  sin(th)];
```

which we transform to points on the perimeter of the ellipse

```
>> x = (sqrtm(E) * y)';
>> plot(x(:,1), x(:,2));
```

which is encapsulated in the Toolbox function

```
>> plot_ellipse(E, [0 0])
```

An ellipsoid is described by a positive-definite symmetric 3×3 matrix. Drawing an ellipsoid is tackled in an analogous fashion and `plot_ellipse` is also able to display a 3-dimensional ellipsoid.

C.1.4.3 Fitting an Ellipse to Data

From a Set of Interior Points

We wish to find the equation of an ellipse that best fits a set of points that lie within the ellipse boundary. A common approach is to find the ellipse that has the same mass properties as the set of points. From the set of N points $\boldsymbol{x}_i = (x_i, y_i)$ we can compute the moments

$$m_{00} = N$$

$$m_{10} = \sum_{i=1}^{N} x_i$$

$$m_{01} = \sum_{i=1}^{N} y_i$$

The center of the ellipse is taken to be the centroid of the set of points

$$(x_c, y_c) = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right)$$

which allows us to compute the central second moments

$$\mu_{20} = \sum_{i=1}^{N} (x_i - x_c)^2$$
$$\mu_{02} = \sum_{i=1}^{N} (y_i - y_c)^2$$
$$\mu_{11} = \sum_{i=1}^{N} (x_i - x_c)(y_i - y_c)$$

The inertia matrix for a general ellipse is the symmetric matrix

$$\boldsymbol{J} = \begin{pmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{pmatrix}$$

where the diagonal terms are the moments of inertia and the off-diagonal terms are the products of inertia. Inertia can be computed more directly by

$$\boldsymbol{J} = \sum_{i=1}^{N} (\boldsymbol{x} - \boldsymbol{x}_c)(\boldsymbol{x} - \boldsymbol{x}_c)^T$$

The relationship between the inertia matrix and the symmetric ellipse matrix is

$$\boldsymbol{E} = \frac{4}{m_{00}} \boldsymbol{J}$$

To demonstrate this we can create a set of points that lie within the ellipse used in the example above

```
% generate a set of points within the ellipse
p = [];
while true
    x = (rand(2,1)-0.5)*4;
    if norm(x'*inv(E)*x) <= 1
        p = [p x];
    end
    if numcols(p) >= 500
        break;
    end
end
plot(p(1,:), p(2,:), '.')

% compute the moments
m00 = mpq_point(p, 0,0);
m10 = mpq_point(p, 1,0);
m01 = mpq_point(p, 0,1);
xc = m10/m00; yc = m01/m00;

% compute second moments relative to centroid
pp = bsxfun(@minus, p, [xc; yc]);

m20 = mpq_point(pp, 2,0);
m02 = mpq_point(pp, 0,2);
m11 = mpq_point(pp, 1,1);

% compute the moments and ellipse matrix
J = [m20 m11; m11 m02];
E_est = 4 * J / m00
```

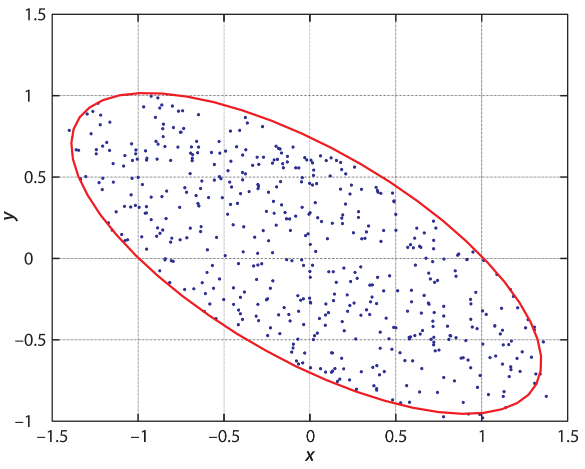


Fig. C.4. Data points (*blue*) with a fitted ellipse (*red*).

which results in an estimate

```
>> E_est
E_est =
    1.8706   -0.9151
   -0.9151    0.9716
```

which is similar to the original value of `E`. The point data is shown in Fig. C.4. We can overlay the estimated ellipse on the point data

```
>> plot_ellipse(E_est, [xc yc],  'r')
```

and the result is shown in red in Fig. C.4.

From a Set of Boundary Points

We wish to find the equation of an ellipse given a set of points (x_i, y_i) that define the boundary of an ellipse. Using the polynomial form of the ellipse Eq. C.5 for each point we write this in matrix form

$$\begin{pmatrix} y_1^2 & x_1 y_1 & x_1 & y_1 & 1 \\ y_2^2 & x_2 y_2 & x_2 & y_2 & 1 \\ & & \vdots & & \\ y_N^2 & x_N y_N & x_N & y_N & 1 \end{pmatrix} \begin{pmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \\ E_5 \end{pmatrix} = \begin{pmatrix} -x_1^2 \\ -x_2^2 \\ \vdots \\ -x_N^2 \end{pmatrix}$$

and for $N \geq 5$ we can solve for the ellipse parameter vector.

C.2 Homogeneous Coordinates

A point in homogeneous coordinates, or the projective space $\mathbb{P}^n$, is represented by a coordinate vector $\tilde{x} = (\tilde{x}_1, \tilde{x}_2 \cdots \tilde{x}_{n+1})$. The Euclidean coordinates are related to the projective coordinates by

$$x_i = \frac{\tilde{x}_i}{\tilde{x}_{n+1}}, \quad i = 1 \cdots n$$

Conversely a homogeneous coordinate vector can be constructed from a Euclidean coordinate vector by

$$\tilde{x} = \left(x_1, x_2 \cdots x_n, 1\right)$$

and the tilde is used to indicate that the quantity is homogeneous.

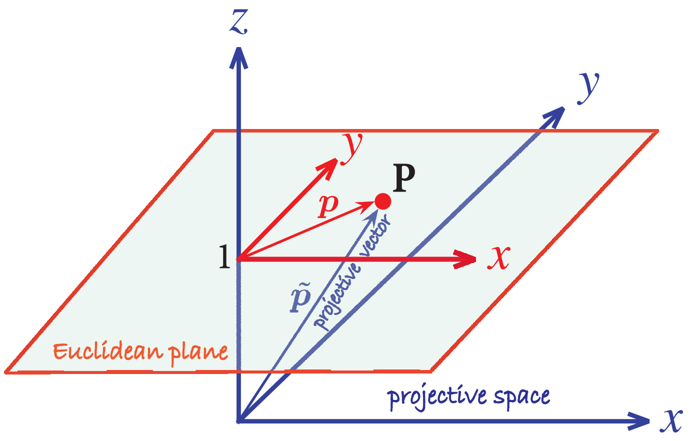


Fig. C.5.
A point P on the Euclidean plane $\mathbb{R}^2$ (*red*) is described by a coordinate vector $\boldsymbol{p} \in \mathbb{R}^2$ which is equivalent to the three-dimensional vector in the projective space $\mathbb{P}^2$ (*blue*) which is the homogeneous coordinate $\tilde{\boldsymbol{p}} \in \mathbb{P}^2$

The extra *degree of freedom* offered by projective coordinates has several advantages. It allows points and lines at infinity, known as ideal points and lines, to be represented using only real numbers. It also means that scale is unimportant, that is $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}' = \alpha\tilde{\boldsymbol{x}}$ both represent the same Euclidean point for all $\alpha \neq 0$. We express this as $\tilde{\boldsymbol{x}} \simeq \tilde{\boldsymbol{x}}'$. Points in homogeneous form can also be rotated with respect to a coordinate frame and translated simply by multiplying the homogeneous coordinate by an $(n+1) \times (n+1)$ homogeneous transformation matrix.

Homogeneous vectors are important in computer vision when we consider points and lines that exist in a plane – a camera's image plane. We can also consider that the homogeneous form represents a ray in Euclidean space as shown in Fig. C.5. The relationship between points and rays is at the core of the projective transformation.

C.2.1 Two Dimensions

In two dimensions there is a duality between points and lines. In $\mathbb{P}^2$ a line is defined by a 3-tuple, $\tilde{\ell} = (\ell_1, \ell_2, \ell_3)^T$, not all zero, and the equation of the line is the set of all points

$$\tilde{\ell}^T \tilde{\boldsymbol{x}} = 0$$

which expands to $\ell_1 x + \ell_2 y + \ell_3 = 0$ and can be manipulated into the more familiar representation of a line. Note that this form can represent a vertical line, parallel to the y-axis, which the familiar form $y = mx + c$ cannot. This is the point equation of a line. The nonhomogeneous vector (ℓ_1, ℓ_2) is a normal to the line, and $(-\ell_2, \ell_1)$ is parallel to the line.

A point is defined by the intersection of two lines. If we write the point equations for two lines $\tilde{\ell}_1^T \tilde{\boldsymbol{p}} = 0$ and $\tilde{\ell}_2^T \tilde{\boldsymbol{p}} = 0$ their intersection is the point with coordinates

$$\tilde{\boldsymbol{p}} = \tilde{\ell}_1 \times \tilde{\ell}_2$$

and is known as the line equation of a point. Similarly, a line joining two points $\tilde{\boldsymbol{p}}_1$ and $\tilde{\boldsymbol{p}}_2$ is given by the cross-product

$$\tilde{\ell}_{12} = \tilde{\boldsymbol{p}}_1 \times \tilde{\boldsymbol{p}}_2$$

Consider the case of two parallel lines at 45° to the horizontal axis

```
>> l1 = [1 -1 0]';
>> l2 = [1 -1 -1]';
```

which we can plot

```
>> plot_homline(l1, 'b')
>> plot_homline(l2, 'r')
```

The intersection point of these parallel lines is

```
>> cross(l1, l2)
ans =
     1     1     0
```

This is an *ideal point* since the third coordinate is zero – the equivalent Euclidean point would be at infinity. Projective coordinates allow points and lines at infinity to be simply represented and manipulated without special logic.

The distance from a point with coordinates $\tilde{\boldsymbol{p}}$ to a line $\tilde{\ell}$ is

$$d = \frac{\tilde{\ell}^T \tilde{\boldsymbol{p}}}{p_3 \sqrt{\ell_1^2 + \ell_2^2}} \tag{C.6}$$

C.2.1.1 Conics

Conic sections are an important family of planar curves that includes circles, ellipses, parabolas and hyperbolas which can be described by

$$Au^2 + Buv + Cv^2 + Du + Ev + F = 0$$

or more concisely as $\tilde{\boldsymbol{p}}^T \boldsymbol{c} \tilde{\boldsymbol{p}} = 0$ where $\boldsymbol{c}$ is a matrix

$$\boldsymbol{c} = \left(\begin{array}{cc|c} A & B/2 & D/2 \\ B/2 & C & E/2 \\ \hline D/2 & E/2 & F \end{array}\right)$$

The determinant of the top-left submatrix indicates the type of conic: negative for a hyperbola, 0 for a parabola and positive for an ellipse.

C.2.2 Three Dimensions

In three dimensions there is a duality between points and planes.

C.2.2.1 Lines

Using the homogeneous representation of the two points $\tilde{\boldsymbol{p}}$ and $\tilde{\boldsymbol{q}}$ we can form a 4×4 skew-symmetric matrix

$$\begin{aligned} \boldsymbol{L} &= \tilde{\boldsymbol{q}} \tilde{\boldsymbol{p}}^T - \tilde{\boldsymbol{p}} \tilde{\boldsymbol{q}}^T \\ &= \begin{pmatrix} 0 & v_3 & -v_2 & -\omega_1 \\ -v_3 & 0 & v_1 & -\omega_2 \\ v_2 & -v_1 & 0 & -\omega_3 \\ \omega_1 & \omega_2 & \omega_3 & 0 \end{pmatrix} \end{aligned}$$

whose 6 unique elements comprise the Plücker coordinate vector. This matrix is rank 2 and the determinant is a quadratic in the Plücker coordinates – a 4-dimensional quadric hypersurface known as the Klein quadric. All points that lie on this manifold are valid lines. Many of the relationships in Sect. C.1.2.2 (between lines and points and planes) can be expressed in terms of this matrix. This matrix is returned by the `L` method of the `Plucker` class.

For a perspective camera with a camera matrix $\boldsymbol{C}$ the 3-dimensional Plücker line represented as a 4×4 skew-symmetric matrix $\boldsymbol{L}$ is projected onto the image plane as

$$\ell = \boldsymbol{C}\boldsymbol{L}\boldsymbol{C}^T$$

which is a homogeneous line in $\mathbb{P}^2$. This is computed automatically if a `Plucker` object is passed to the `project` method of a `CentralCamera` object.

C.2.2.2 Planes

The plane described by $\pi\tilde{\boldsymbol{x}} = 0$ can be defined by a line and a point

$$\pi = \boldsymbol{L}\tilde{\boldsymbol{p}}$$

The join and incidence relationships are more complex than the cross products used for the 2-dimensional case. Three points define a plane and the join relationship is

$$\begin{pmatrix} \tilde{\boldsymbol{p}}_1^T \\ \tilde{\boldsymbol{p}}_2^T \\ \tilde{\boldsymbol{p}}_3^T \end{pmatrix} \tilde{\pi} = 0$$

and the solution is found from the right-null space of the matrix. The incidence of three planes is the dual

$$\begin{pmatrix} \tilde{\pi}_1^T \\ \tilde{\pi}_2^T \\ \tilde{\pi}_3^T \end{pmatrix} \tilde{\boldsymbol{p}} = 0$$

and is an ideal point, zero last component, if the planes do not intersect at a point.

C.2.2.3 Quadrics

Quadrics, short for quadratic surfaces, are a rich family of 3-dimensional surfaces. There are 17 standard types including spheres, ellipsoids, hyperboloids, paraboloids, cylinders and cones all described by

$$\tilde{\boldsymbol{x}}^T \boldsymbol{Q} \tilde{\boldsymbol{x}} = 0$$

where $\boldsymbol{Q} \in \mathbb{R}^{4\times 4}$ is symmetric.

For a perspective camera with a camera matrix $\boldsymbol{C}$ the outline of the quadric is projected to the image plane by

$$\boldsymbol{c}^* = \boldsymbol{C}\boldsymbol{Q}^*\boldsymbol{C}^T$$

where $\boldsymbol{c}$ is a 3×3 matrix describing the conic, see Sect. C.2.1.1, and $(\cdot)^*$ represents the adjugate operation, see Appendix B.

C.3 Geometric Transformations

A linear transform is $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ and an affine transform is

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \tag{C.7}$$

which comprises a linear transformation *and* a change of origin. Examples of affine transformations include translation, scaling, homothety, similarity transformation, reflection, rotation, shear mapping, and compositions of them in any combination and sequence. Every linear transformation is affine, but not every affine transformation is linear.

In homogeneous coordinates we can write Eq. C.7 as

$$\tilde{\boldsymbol{y}} = \boldsymbol{H}\tilde{\boldsymbol{x}}, \quad \text{where } \boldsymbol{H} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{b} \\ 0 & 1 \end{pmatrix}$$

and the transformation operates on a point with homogeneous coordinates $\tilde{\boldsymbol{x}}$. If a vector is defined as the difference between two homogeneous points $\tilde{\boldsymbol{p}}$ and $\tilde{\boldsymbol{q}}$ then the difference $\tilde{\boldsymbol{p}} - \tilde{\boldsymbol{q}}$ is a 4-vector whose last element is zero, distinguishing a point from a vector.

Affine space is a generalization of Euclidean space and has no distinguished point that serves as an origin. Hence, no vector has a fixed origin and no vector can be uniquely associated to a point an affine space, there are instead displacement vectors between two points of the space. Thus it makes sense to subtract two points of the space, giving a vector, but it does not make sense to add two points of the space. Likewise, it makes sense to add a vector to a point of an affine space, resulting in a new point displaced from the starting point by that vector.

In two-dimensions the most general transformation is projective transformation projective, also known as a collineation

$$\boldsymbol{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix}$$

which is unique up to scale and one element has been normalized to one. It has 8 degrees of freedom.

The affine transformation is a subset where the elements of the last row are fixed

$$\boldsymbol{H} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

and has 6 degrees of freedom.

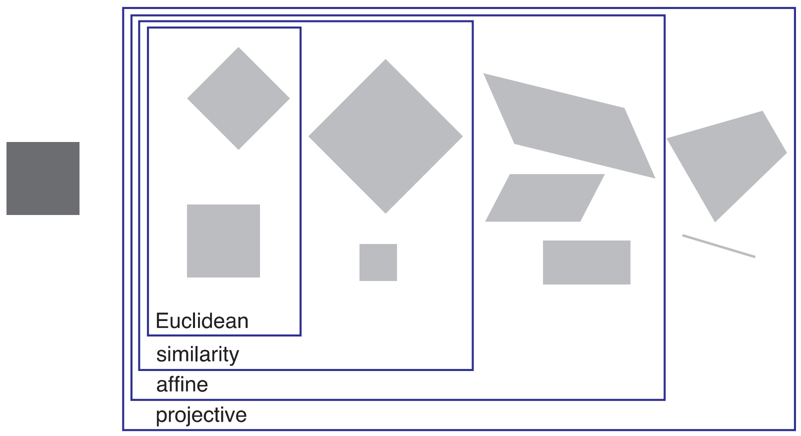


Fig. C.6. A 2-dimensional square (*dark grey*) is operated on by various transformations from the most limited (Euclidean) to the most general (projective)

Table C.1. For various planar transformation families the possible geometric transformations and the geometric properties which are preserved are listed

	Euclidean	Similarity	Affine	Projective
Geometric transformation				
Rotation	✓	✓	✓	✓
Translation	✓	✓	✓	✓
Reflection		✓	✓	✓
Uniform scaling		✓	✓	✓
Nonuniform scaling			✓	✓
Shear			✓	✓
Perspective projection				✓
Preserved geometric properties (invariants)				
Length	✓			
Angle	✓	✓		
Ratio of lengths	✓	✓		
Parallelism	✓	✓	✓	
Incidence	✓	✓	✓	✓
Cross ratio	✓	✓	✓	✓

The similarity transformation is further subset

$$\boldsymbol{H} = \begin{pmatrix} s\boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

where $\boldsymbol{R} \in \mathbf{SO}(2)$ resulting in only 4 degrees of freedom. Similarity transforms, without reflection, are sometimes referred to as a Procrustes transform.

Finally the Euclidean or rigid-body transformation

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \in \mathbf{SE}(2)$$

is the most restrictive and has only 3 degrees of freedom. Some graphical examples of the effect of the various transformations on a square are shown in Fig. C.6. The possible geometric transformations for each type of transform are summarized in Table C.1 along with the geometric properties which are unchanged, or invariant, under that transformation. We see that while Euclidean is most restrictive in terms of the geometric transformations it can perform it is able to preserve important properties such as length and angle.

D Lie Groups and Algebras

We cannot go very far in the study of rotations or rigid-body motion without coming across the terms Lie groups, Lie algebras or Lie brackets – all named in honor of the Norwegian mathematician Sophus Lie. Rotations and rigid-body motion in 2- and 3-dimensions can be represented by matrices which form Lie groups and which have Lie algebras.

We will start simply by considering the set of all real 2×2 matrices $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

which we could write as a linear combination of basis matrices

$$\boldsymbol{A} = a_{11}\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + a_{12}\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + a_{21}\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + a_{22}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

where each basis matrix represents a *direction* in a 4-dimensional space of 2×2 matrices. That is, the four axes of this space are *parallel* with each of these basis matrices. Any 2×2 matrix can be represented by a point in this space – this particular matrix is a point with the coordinates $(a_{11}, a_{12}, a_{21}, a_{22})$.

All proper rotation matrices, those belonging to $\mathbf{SO}(2)$, are a subset of points within the space of all 2×2 matrices. For this example the points lie in a 1-dimensional subset, a closed curve, in the 4-dimensional space. This is an instance of a manifold, a lower-dimensional smooth *surface* embedded within a space.

The notion of a curve in the 4-dimensional space makes sense when we consider that the $\mathbf{SO}(2)$ rotation matrix

$$\boldsymbol{A} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

has only one free parameter, and varying that parameter moves the point along the manifold.

Sophus Lie (1842–1899) (surname pronounced lee) was a Norwegian mathematician who obtained his Ph.D. from the University of Christiania in Oslo in 1871. He spent time in Berlin working with Felix Klein, and later contributed to Klein's Erlangan program to characterize geometries based on group theory and projective geometry. On a visit to Milan during the Franco-Prussian war he was arrested as a German spy and spent one month in prison. He is best known for his discovery that continuous transformation groups (now called Lie groups) can be understood by linearizing them and studying their generating vector spaces. He is buried in the Vår Frelsers gravlund in Oslo. (Photograph by Ludwik Szacinski)

Invoking mathematical formalism we say that rotations $\mathbf{SO}(2)$ and $\mathbf{SO}(3)$, and rigid-body motions $\mathbf{SE}(2)$ and $\mathbf{SE}(3)$ are matrix Lie groups and this has two implications. Firstly, they are an *algebraic group*, a mathematical structure comprising elements and a single operator. In simple terms, a group $\mathbf{G}$ has the following properties:

1. if g_1 and g_2 are elements of the group, that is $g_1, g_2 \in \mathbf{G}$, then the result of the group's operator $\circ$ is also an element of the group: $g_1 \circ g_2 \in \mathbf{G}$. In general, groups are not commutative so $g_1 \circ g_2 \neq g_2 \circ g_1$. For rotations and rigid-body motions the group operator $\circ$ represents composition.
2. the group operator is associative, that is, $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.
3. for $g \in \mathbf{G}$ there is an identity element $I \in \mathbf{G}$ such that $g \circ I = I \circ g = g$.
4. for every $g \in \mathbf{G}$ there is a unique inverse $h \in \mathbf{G}$ such that $g \circ h = h \circ g = I$.

In this book's notation the $\oplus$ operator is the group operator.

In this book's notation the identity is denoted by 0 (implying null motion) so we can say that $\xi \oplus 0 = 0 \oplus \xi = \xi$.

In this book's notation we use the operator $\ominus\xi$ to form the inverse.

The second implication of being a Lie group is that there is a smooth (differentiable) manifold structure. At any point on the manifold we can construct tangent vectors. The set of all tangent vectors at that point form a vector space – the tangent space. This is the multidimensional equivalent to a tangent line on a curve, or a tangent plane on a solid. We can think of this as the set of all possible derivatives of the manifold at that point.

The tangent space *at the identity* is described by the Lie algebra of the group, and the basis directions of the tangent space are called the generators of the group. Points in this tangent space map to elements of the group via the exponential function. If $\mathbf{g}$ is the Lie algebra for group $\mathbf{G}$ then

$$\forall X \in \mathbf{g} \Rightarrow \mathrm{e}^{X} \in \mathbf{G}$$

where the elements of $\mathbf{g}$ and $\mathbf{G}$ are matrices of the same size and which each have a specific structure.

The surface of a sphere is a manifold in 3-dimensional space and at any point on that surface we can create a tangent vector. In fact we can create an infinite number of them and they lie within a plane which is a 2-dimensional vector space – the tangent space. We can choose a set of basis directions and establish a 2-dimensional coordinate system and we can map points on the plane to points on the sphere's surface.

Now consider an arbitrary real 3×3 matrix $\boldsymbol{A} \in \mathbb{R}^{3\times3}$

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

which we could write as a linear combination of basis matrices

$$\boldsymbol{A} = a_{11}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + a_{12}\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \cdots + a_{33}\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

where each basis matrix represents a *direction* in a 9-dimensional space of 3×3 matrices. Every possible 3×3 matrix is represented by a point in this space.

Not all matrices in this space are proper rotation matrices belonging to $\mathbf{SO}(3)$, but those that do lie on a manifold since $\mathbf{SO}(3)$ is a Lie group. The null rotation, represented by the identity matrix, is one point in this space. At that point we can construct a tangent space which has only 3 dimensions. Every point in the tangent space – the derivatives of the manifold – can be expressed as a linear combination of basis matrices

$$\Omega = \omega_1 \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}}_{G_1} + \omega_2 \underbrace{\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}}_{G_2} + \omega_3 \underbrace{\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{G_3} \tag{D.1}$$

The equivalent algebra is denoted using lower case letters and is a set of matrices.

which is the Lie algebra of the **SO**(3) group. The bases of this space: G_1, G_2 and G_3 are called the generators of **SO**(3) and belong to **so**(3).◀

Equation D.1 can be written as a skew-symmetric matrix parameterized by the vector $\omega = (\omega_1, \omega_2, \omega_3) \in \mathbb{R}^3$

$$\Omega = [\omega]_\times = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \in \mathbf{so}(3)$$

and this reflects the 3 degrees of freedom of the **SO**(3) group embedded in the space of all 3×3 matrices. The 3DOF is consistent with our intuition about rotations in 3D space and also Euler's rotation theorem.

Mapping between vectors and skew-symmetric matrices is frequently required and the following shorthand notation will be used

$$[\cdot]_\times : \mathbb{R} \mapsto \mathbf{so}(2),\ \mathbb{R}^3 \mapsto \mathbf{so}(3)$$
$$\vee_\times(\cdot) : \mathbf{so}(2) \mapsto \mathbb{R},\ \mathbf{so}(3) \mapsto \mathbb{R}^3$$

The first mapping is performed by the Toolbox function `skew` and the second by `vex` (which is named after the $\vee_\times$).

The exponential of *any* matrix in **so**(3) is a valid member of **SO**(3)

$$R(\theta, \hat{\omega}) = \mathrm{e}^{[\hat{\omega}]_\times \theta} \in \mathbf{SO}(3)$$

and an efficient closed-form solution is given by Rodrigues' rotation formula

$$R(\theta, \hat{\omega}) = I + \sin\theta [\hat{\omega}]_\times + (1 - \cos\theta)[\hat{\omega}]_\times^2$$

Finally, consider an arbitrary real 4×4 matrix $A \in \mathbb{R}^{4\times4}$

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

which we could write as a linear combination of basis matrices

$$A = a_{11} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + a_{12} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \cdots + a_{44} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where each basis matrix represents a *direction* in a 16-dimensional space of all possible 4×4 matrices. Every 4×4 matrix is represented by a point in this space.

Not all matrices in this space are proper homogeneous transformation matrices belonging to **SE**(3), but those that do lie on a smooth manifold. The null motion (zero rotation and translation), which is represented by the identity matrix, is one point in this space. At that point we can construct a tangent space, which has 6 dimensions in this case, and points in the tangent space can be expressed as a linear combination of basis matrices

$$\boldsymbol{\Sigma} = \omega_1 \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \omega_2 \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + \omega_3 \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$
$$+ v_1 \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + v_2 \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} + v_3 \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

and these generator matrices belong to the Lie algebra of the group **SE**(3) and are denoted **se**(3). This can be written in general form as

$$\boldsymbol{\Sigma} = [S] = \left(\begin{array}{ccc|c} 0 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 0 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 0 & v_3 \\ \hline 0 & 0 & 0 & 0 \end{array}\right) \in \mathbf{se}(3)$$

which is an augmented skew symmetric matrix parameterized by $S = (\boldsymbol{v}, \boldsymbol{\omega}) \in \mathbb{R}^6$ which is referred to as a twist and has physical interpretation in terms of a screw axis direction and position. The sparse matrix structure and this concise parameterization reflects the 6 degrees of freedom of the **SE**(3) group embedded in the space of all 4×4 matrices. We extend our earlier shorthand notation

$$[\cdot]\colon \mathbb{R}^3 \mapsto \mathbf{se}(2),\ \mathbb{R}^6 \mapsto \mathbf{se}(3)$$
$$\vee(\cdot)\colon \mathbf{se}(2) \mapsto \mathbb{R}^3,\ \mathbf{se}(3) \mapsto \mathbb{R}^6$$

We can use these operators to convert between a twist representation which is a 6-vector and a Lie algebra representation which is a 4×4 augmented skew-symmetric matrix. We convert the Lie algebra to the Lie group representation using

$$\boldsymbol{T}(\theta, S) = \mathrm{e}^{[S]\theta} \in \mathbf{SE}(3)$$

or the inverse using the matrix logarithm. The exponential and the logarithm each have an efficient closed form solution.

Transforming a Twist – the Adjoint Representation

We have seen that rigid-body motions can be described by a twist which represents motion in terms of a screw axis direction and position, for example in Fig. D.1 the twist S_A can be used to transform points on the body. If the screw is rigidly attached to the body which undergoes some motion in **SE**(3) the new twist is

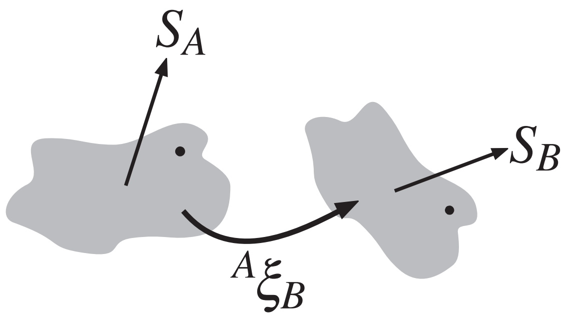


Fig. D.1.
Points in the body (*grey cloud*) can be transformed by the twist S_A. If the body and the screw axis undergo a rigid-body transformation $^A\xi_B$ the new twist is S_B

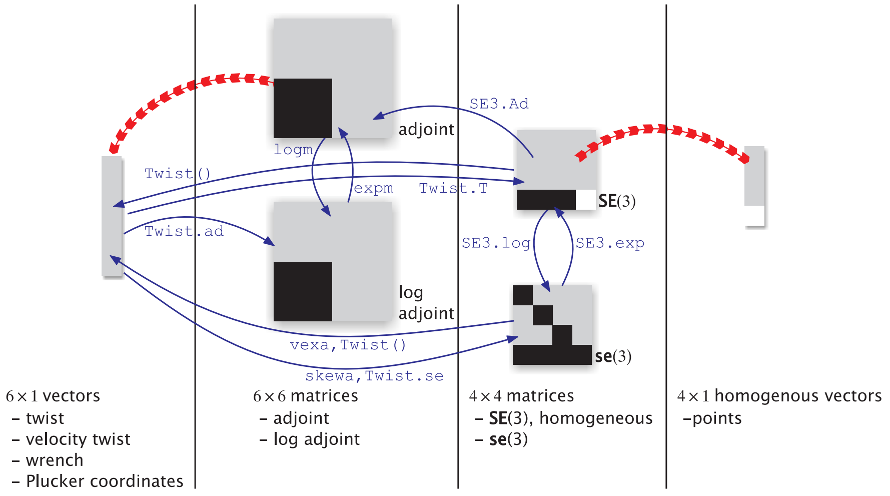


Fig. D.2. The menagerie of **SE**(3) related quantities. Matrix values are coded as: 0 (*black*), 1 (*white*), other values (*grey*). Transformations between types are indicated by *blue arrows* with the relevant class plus method name. Operations are indicated by *red arrows*: the tail-end object operates on the head-end object and results in another object of the head-end type

$$^{B}S = \mathrm{Ad}\left({}^{B}\xi_{A}\right){}^{A}S$$

where

$$\mathrm{Ad}(\xi) = \begin{pmatrix} \boldsymbol{R} & [\boldsymbol{t}]_\times \boldsymbol{R} \\ 0 & \boldsymbol{R} \end{pmatrix} \in \mathbb{R}^{6\times 6} \tag{D.2}$$

is the adjoint representation of the rigid-body motion. Alternatively we can write

$$\mathrm{Ad}\left(\mathrm{e}^{[S]}\right) = \mathrm{e}^{\mathrm{ad}(S)}$$

where ad(S) is the logarithm of the adjoint and defined in terms of the twist parameters as

$$\mathrm{ad}(S) = \begin{pmatrix} [\boldsymbol{\omega}]_\times & [\boldsymbol{v}]_\times \\ 0 & [\boldsymbol{\omega}]_\times \end{pmatrix} \in \mathbb{R}^{6\times 6}$$

The relationship between the various mathematical objects discussed are shown in Fig. D.2.

E Linearization, Jacobians and Hessians

In robotics and computer vision the equations we encounter are often nonlinear. To apply familiar and powerful analytic techniques we must work with linear or quadratic approximations to these equations. The principle is illustrated in Fig. E.1 for the 1-dimensional case, and the analytical approximations shown in red are made at $x = x_0$. The approximation equals the nonlinear function at x_0 but is increasing inaccurate as we move away from that point. This is called a local approximation since it is valid in a region local to x_0 – the size of the valid region depends on the severity of the nonlinearity. This approach can be extended to an arbitrary number of dimensions.

Scalar Function of a Scalar

The function f: $\mathbb{R} \mapsto \mathbb{R}$ can be expressed as a Taylor series

$$f(x_0 + \Delta) = f(x_0) + \frac{\mathrm{d}f}{\mathrm{d}x}\Delta + \tfrac{1}{2}\frac{\mathrm{d}^2 f}{\mathrm{d}x^2}\Delta^2 + \cdots$$

which we truncate to form a first-order or linear approximation

$$f'(\Delta) \approx f(x_0) + J(x_0)\Delta$$

or a second-order approximation

$$f'(\Delta) \approx f(x_0) + J(x_0)\Delta + \tfrac{1}{2}H(x_0)\Delta^2$$

where $\Delta \in \mathbb{R}$ is an infinitesimal change in x relative to the linearization point x_0, and the first and second derivatives are given by $J(x_0) = \mathrm{d}f/\mathrm{d}x|_{x_0}$ and $H(x_0) = \mathrm{d}^2 f/\mathrm{d}x^2|_{x_0}$ respectively.

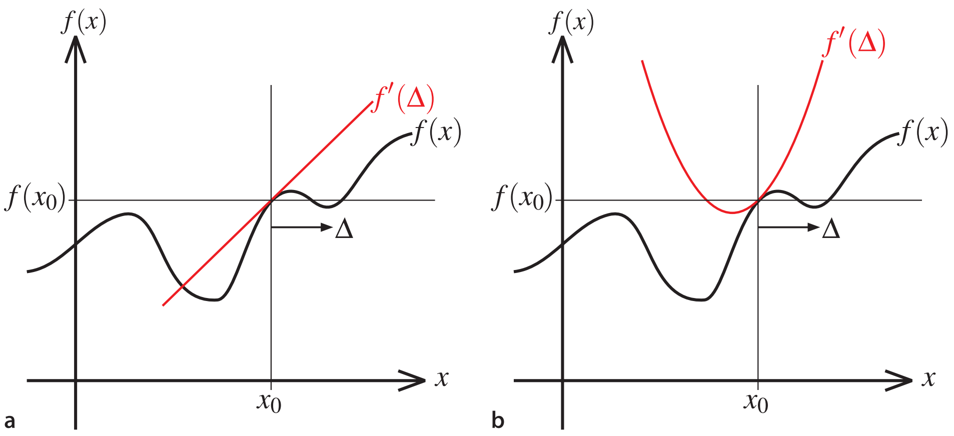


Fig. E.1. The nonlinear function $f(x)$ (*black*) is approximated (*red*) at the point $x = x_0$ by **a** a line – a linear or first-order approximation, **b** a parabola – a second-order approximation. At the linearization point both curves are equal and tangent to the function while for **b** the second derivatives also match

Scalar Function of a Vector

The scalar field $f(\boldsymbol{x})\colon \mathbb{R}^n \mapsto \mathbb{R}$ can be expressed as a Taylor series

$$f(\boldsymbol{x}_0 + \boldsymbol{\Delta}) = f(\boldsymbol{x}_0) + \boldsymbol{J}(\boldsymbol{x}_0)\boldsymbol{\Delta} + \tfrac{1}{2}\boldsymbol{\Delta}^T \boldsymbol{H}(\boldsymbol{x}_0)\boldsymbol{\Delta} + \cdots$$

which we can truncate to form a first-order or linear approximation

$$f'(\boldsymbol{\Delta}) \approx f(\boldsymbol{x}_0) + \boldsymbol{J}(\boldsymbol{x}_0)\boldsymbol{\Delta}$$

or a second-order approximation

$$f'(\boldsymbol{\Delta}) \approx f(\boldsymbol{x}_0) + \boldsymbol{J}(\boldsymbol{x}_0)\boldsymbol{\Delta} + \tfrac{1}{2}\boldsymbol{\Delta}^T \boldsymbol{H}(\boldsymbol{x}_0)\boldsymbol{\Delta}$$

where $\boldsymbol{\Delta} \in \mathbb{R}^n$ is an infinitesimal change in $\boldsymbol{x} \in \mathbb{R}^n$ relative to the linearization point $\boldsymbol{x}_0$, $\boldsymbol{J} \in \mathbb{R}^{1\times n}$ is the vector version of the first derivative, and $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ is the Hessian – the matrix version of the second derivative.

The derivative of the function $f(\cdot)$ with respect to the vector $\boldsymbol{x}$ is

$$\boldsymbol{J}(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}$$

and is itself a vector that points in the direction at which the function $f(\boldsymbol{x})$ has maximal increase. It is often written as $\nabla_x f$ to make explicit that the differentiation is with respect to $\boldsymbol{x}$.

The Hessian is an $n \times n$ symmetric matrix of second derivatives

$$\boldsymbol{H} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \frac{\partial^2 f}{\partial x_2 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

The function is at a critical point when the Jacobian is not full rank. If the Hessian is positive definite then the function is at a local minimum, if negative definite then a local maximum, and if indefinite then the function is at a saddle point.

For functions which are quadratic in $\boldsymbol{x}$, as is the case for least-squares problems, it can be shown that the Hessian is

$$\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{J}(\boldsymbol{x})^T \boldsymbol{J}(\boldsymbol{x}) + \sum_{i=1}^{m} f_i(\boldsymbol{x}) \frac{\partial^2 f_i}{\partial \boldsymbol{x}^2} \approx \boldsymbol{J}(\boldsymbol{x})^T \boldsymbol{J}(\boldsymbol{x})$$

which is frequently approximated by just the first term and this is key to Gauss-Newton least-squares optimization discussed in Sect. F.2.2.

Ludwig Otto Hesse (1811–1874) was a German mathematician, born in Königsberg, Prussia, who studied under Jacobi (p. 232) and Bessel at the University of Königsberg. He taught at Königsberg, Halle, Heidelberg and finally at the newly established Polytechnic School in Munich. In 1869 he joined the Bavarian Academy of Sciences.

Vector Function of a Vector

The vector field $\boldsymbol{f}(\boldsymbol{x})\colon \mathbb{R}^n \mapsto \mathbb{R}^m$ can be expressed as a Taylor series which can also be written as

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{pmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \\ \vdots \\ f_n(\boldsymbol{x}) \end{pmatrix}$$

where $f_i: \mathbb{R}^m \to \mathbb{R}$ for $i \in \{1, 2, \cdots n\}$. The derivative of $\boldsymbol{f}$ with respect to the vector $\boldsymbol{x}$ can be expressed in matrix form as a Jacobian matrix

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

which can also be written as

$$\boldsymbol{J}(\boldsymbol{x}) = \begin{pmatrix} \nabla f_1^T \\ \nabla f_2^T \\ \vdots \\ \nabla f_n^T \end{pmatrix}$$

This derivative is also known as the tangent map of $\boldsymbol{f}$, denoted $T\boldsymbol{f}$, or the differential of $\boldsymbol{f}$ denoted $D\boldsymbol{f}$. To make explicit that the differentiation is with respect to $\boldsymbol{x}$ this can be denoted as $\boldsymbol{J}_x$, $T_x\boldsymbol{f}$, $D_x\boldsymbol{f}$ or even $\partial \boldsymbol{f} / \partial \boldsymbol{x}$.

The Hessian in this case is $\boldsymbol{H} \in \mathbb{R}^{n \times m \times n}$ which is a 3-dimensional array called a cubix.

Deriving Jacobians

Jacobians of functions are required for many optimization algorithms as well as for the extended Kalman filter, and can be evaluated numerically or symbolically.

Consider Eq. 6.8 for the range and bearing angle of a landmark given the pose of the vehicle and the position of the landmark. We can express this as the very simple MATLAB® anonymous function

```
>> zrange = @(xi, xv, w) ...
       [ sqrt((xi(1)-xv(1))^2 + (xi(2)-xv(2))^2) + w(1);
         atan((xi(2)-xv(2))/(xi(1)-xv(1)))-xv(3) + w(2) ];
```

To estimate the Jacobian $\boldsymbol{H}_{xv} = \partial \boldsymbol{h} / \partial \boldsymbol{x}_v$ for $\boldsymbol{x}_v = (1, 2, \frac{\pi}{3})$ and $\boldsymbol{x}_i = (10, 8)$ we can compute a first-order numerical difference

```
>> xv = [1, 2, pi/3]; xi = [10, 8]; w= [0,0];
>> h0 = zrange(xi, xv, w)
h0 =
   10.8167
   -0.4592
>> d = 0.001;
>> J = [ zrange(xi, xv+[1,0,0]*d, w)-h0 ...
         zrange(xi, xv+[0,1,0]*d, w)-h0, ...
         zrange(xi, xv+[0,0,1]*d,w)-h0]  /  d
J =
   -0.8320   -0.5547         0
    0.0513   -0.0769   -1.0000
```

which shares the characteristic last column with the Jacobian shown in Eq. 6.14. Note that in computing this Jacobian we have set the measurement noise `w` to zero. The principal difficulty with this approach is choosing `d`, the difference used to compute the finite-difference approximation to the derivative. Too large and the results will be quite inaccurate if the function is nonlinear, too small and numerical problems will lead to reduced accuracy.

Alternatively we can perform the differentiation symbolically. This particular function is relatively simple and the derivatives can be determined easily using differential calculus. The numerical derivative can be used as a quick check for correctness. To avoid the possibility of error, or for more complex functions we can perform the differentiation symbolically using any of a large number of computer algebra packages. Using the MATLAB Symbolic Math Toolbox™ we can declare some symbolic variables

```
>> syms xi yi xv yv thetav wr wb
```

and then evaluate the same function as above

```
>> z = zrange([xi yi], [xv yv thetav], [wr wb])
z =
        wr + ((xi - xv)/(yi - yv)^2)^(1/2)
 wb - thetav + atan((yi - yv)/(xi - xv))
```

which is simply Eq. 6.8 in MATLAB symbolic form. The Jacobian is computed by a Symbolic Math Toolbox™ function

```
>> J = jacobian(z, [xv yv thetav])
J =
[ -(2*xi - 2*xv)/(2*((xi - xv)^2 + (yi - yv)^2)^(1/2)),↵
    -(2*yi - 2*yv)/(2*((xi - xv)^2 + (yi - yv)^2)^(1/2)),  0]
[ (yi - yv)/((xi - xv)^2*((yi - yv)^2/(xi - xv)^2 + 1)),↵
    -1/((xi - xv)*((yi - yv)^2/(xi - xv)^2 + 1)), -1]
```

which has the required dimensions

```
>> about(J)
J [sym] : 2x3 (112 bytes)
```

and the characteristic last column. We could cut and paste this code into our program or automatically create a MATLAB callable function

```
>> Jf = matlabFunction(J);
```

where `Jf` is a MATLAB function handle. We can evaluate the Jacobian at the operating point given above

```
>> xv = [1, 2, pi/3]; xi = [10, 8]; w = [0,0];
>> Jf( xi(1), xv(1), xi(2), xv(2) )
ans =
   -0.8321   -0.5547         0
    0.0513   -0.0769   -1.0000
```

which is similar to the approximation above obtained numerically. The function `matlabFunction` can also write the function to an M-file. The functions `ccode` and `fcode` generate C and Fortran representations of the Jacobian.

Another interesting approach is the package ADOL-C which is an open-source tool for the automatic differentiation of C and C++ programs, that is, given a function written in C it will return a Jacobian function written in C. It is available at http://www.coin-or.org/projects/ADOL-C.xml.

F Solving Systems of Equations

Solving systems of linear and nonlinear equations, particularly over-constrained systems, is a common problem in robotics and computer vision.

F.1 Linear Problems

F.1.1 Nonhomogeneous Systems

These are equations of the form

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

where we wish to solve for the unknown vector $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$ are constants.

If $n = m$ then $\boldsymbol{A}$ is square, and if $\boldsymbol{A}$ is nonsingular then the solution is obtained using the matrix inverse

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$$

In practice we often encounter systems where $m > n$, that is there are more equations than unknowns. In general there will not be an exact solution but we can attempt to find the *best* solution, in a least-squares sense, which is

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|$$

That solution is given by

$$\boldsymbol{x}^* = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{b} = \boldsymbol{A}^+\boldsymbol{b}$$

Since the inverse left multiplies $\boldsymbol{b}$.

which is known as the pseudo inverse or more formally the left-generalized inverse.◂ Using SVD where $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ this is

$$\boldsymbol{x} = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T\boldsymbol{b}$$

where $\boldsymbol{\Sigma}^{-1}$ is simply the element-wise inverse of the diagonal elements of $\boldsymbol{\Sigma}^T$.

If the matrix is singular, or the system is under constrained $n < m$, then there are infinitely many solutions. We can again use the SVD approach

$$\boldsymbol{x} = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T\boldsymbol{b}$$

where this time $\boldsymbol{\Sigma}^{-1}$ is the element-wise inverse of the *nonzero* diagonal elements of $\boldsymbol{\Sigma}$, all other zeros are left in place.

In MATLAB all these problems can be solved using the backslash operator

```
>> x = A\b
```

For the problem

$$RP = Q$$

where R is an unknown rotation matrix in $\mathbf{SO}(n)$, and $P = \{p_1 \cdots p_m\} \in \mathbb{R}^{n \times m}$ and $Q = \{q_1 \cdots q_m\} \in \mathbb{R}^{n \times m}$ comprise column vectors for which $q_i = Rp_i$. We first compute the moment matrix

$$M = \sum_{i=1}^{N} q_i p_i^T$$

and take then compute the SVD $M = U\Sigma V^T$. The least squares estimate of the rotation matrix is

$$R = UV^T$$

and is guaranteed to be an orthogonal matrix.

F.1.2 Homogeneous Systems

These are equations of the form

$$Ax = 0$$

and always have the trivial solution $x = 0$. If A is square and nonsingular this is the only solution. Otherwise, if A is not of full rank, that is the matrix is nonsquare, or square and singular then there are an infinite number of solutions which are linear combinations of vectors in the right null space of A which is computed by the MATLAB function `null`.

F.2 Nonlinear Problems

Many problems in robotics and computer vision involves sets of nonlinear equations. Solution of these problems requires linearizing the equations about an estimated solution, solving for an improved solution and iterating. Linearization is discussed in Appendix E.

F.2.1 Finding Roots

Consider a set of equations expressed in the form

$$f(x) = 0$$

where $f\colon \mathbb{R}^n \mapsto \mathbb{R}^m$. This is a nonlinear version of the homogeneous system described above. We first linearize the equation about our best estimate of the solution x_0

$$f(x_0 + \Delta) = f(x_0) + J(x_0)\Delta + \tfrac{1}{2}\Delta^T H(x_0)\Delta + \cdots \tag{F.1}$$

where $\Delta \in \mathbb{R}^n$ is an infinitesimal change in x relative to x_0. We truncate this to form a linear approximation

$$f'(\Delta) \approx f_0 + J\Delta \tag{F.2}$$

where $\boldsymbol{f}_0 = \boldsymbol{f}(\boldsymbol{x}_0)$ is the function value and $\boldsymbol{J} = \boldsymbol{J}(\boldsymbol{x}_0) \in \mathbb{R}^{1\times n}$ the Jacobian, both evaluated at the linearization point. Now we solve an approximation of our original problem $\boldsymbol{f}'(\Delta) = 0$

$$\boldsymbol{f}_0 + \boldsymbol{J}\boldsymbol{\Delta} = 0 \Rightarrow \boldsymbol{\Delta} = -\boldsymbol{J}^{-1}\boldsymbol{f}_0$$

If $n \neq m$ then $\boldsymbol{J}$ is nonsquare and we can use the pseudo-inverse or the MATLAB backslash operator `-J\f0`. The computed step $\boldsymbol{\Delta}$ is based on an approximation to the original nonlinear function so $\boldsymbol{x}_0 + \boldsymbol{\Delta}$ will generally not be the solution but it will be closer. This leads to an iterative solution – the Newton-Raphson method:

```
1 while $\|\boldsymbol{f}(\boldsymbol{x}_0)\| > \varepsilon$ do
2     compute $\boldsymbol{f}_0 = \boldsymbol{f}(\boldsymbol{x}_0), \boldsymbol{J}(\boldsymbol{x}_0)$
3     $\boldsymbol{\Delta} = -\boldsymbol{J}^{-1}\boldsymbol{f}_0$
4     $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}_0 + \boldsymbol{\Delta}$
5 end
```

F.2.2 Nonlinear Minimization

A very common class of problems involves finding the *minimum* of a scalar function $f(\boldsymbol{x})$: $\mathbb{R}^n \mapsto \mathbb{R}$ which can be expressed as

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

The derivative of the linearized system Eq. F.2 is

$$\frac{\mathrm{d}f'}{\mathrm{d}\boldsymbol{\Delta}} = \boldsymbol{J}$$

and if we consider the function to be a multi-dimensional surface then $\boldsymbol{J}(\boldsymbol{x}_0)$ is vector indicating the direction and magnitude of the *slope* at $\boldsymbol{x} = \boldsymbol{x}_0$ so an update of

$$\boldsymbol{\Delta} = -\beta\boldsymbol{J}$$

will move the estimate *down hill* toward the minimum. This leads to an iterative solution called gradient descent:

```
1 repeat
2 |   compute $\boldsymbol{J} = \boldsymbol{J}(\boldsymbol{x}_0)$
3 |   $\boldsymbol{\Delta} = -\beta\boldsymbol{J}$
4 |   $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}_0 + \boldsymbol{\Delta}$
5 until $\|\boldsymbol{\Delta}\| < \varepsilon$
```

and the challenge is to choose the appropriate step size β.

If we include the second-order term from Eq. F.1 the approximation becomes

$$f'(\boldsymbol{\Delta}) \approx f_0 + \boldsymbol{J}\boldsymbol{\Delta} + \tfrac{1}{2}\boldsymbol{\Delta}^T\boldsymbol{H}(\boldsymbol{x}_0)\boldsymbol{\Delta}$$

and to find its minima we take the derivative and set it to zero

$$\frac{\mathrm{d}f'}{\mathrm{d}\boldsymbol{\Delta}} = 0 \Rightarrow \boldsymbol{J} + \boldsymbol{H}\boldsymbol{\Delta} = 0$$

and the update is

$$\boldsymbol{\Delta} = -\boldsymbol{H}^{-1}\boldsymbol{J}$$

This leads to another iterative solution – Newton's method. The challenge is determining the Hessian of the nonlinear system, either by numerical approximation or symbolic manipulation.

F.2.3 Nonlinear Least Squares Minimization

Very commonly the scalar function we wish to optimize is a quadratic cost function

$$F(\boldsymbol{x}) = \left\|\boldsymbol{f}(\boldsymbol{x})\right\|^2 = \boldsymbol{f}(\boldsymbol{x})^T \boldsymbol{f}(\boldsymbol{x})$$

where $\boldsymbol{f}(\boldsymbol{x})\colon \mathbb{R}^n \mapsto \mathbb{R}^m$ is some vector-valued nonlinear function which we can linearize as

$$\boldsymbol{f}'(\boldsymbol{\Delta}) \approx \boldsymbol{f}_0 + \boldsymbol{J}\boldsymbol{\Delta}$$

and the scalar cost is

$$\begin{aligned} F(\boldsymbol{\Delta}) &\approx \left(\boldsymbol{f}_0 + \boldsymbol{J}\boldsymbol{\Delta}\right)^T \left(\boldsymbol{f}_0 + \boldsymbol{J}\boldsymbol{\Delta}\right) \\ &\approx \boldsymbol{f}_0^T \boldsymbol{f}_0 + \boxed{\boldsymbol{f}_0^T \boldsymbol{J}\boldsymbol{\Delta} + \boldsymbol{\Delta}^T \boldsymbol{J}^T \boldsymbol{f}_0} + \boldsymbol{\Delta}^T \boldsymbol{J}^T \boldsymbol{J}\boldsymbol{\Delta} \\ &\approx \boldsymbol{f}_0^T \boldsymbol{f}_0 + 2\boldsymbol{f}_0^T \boldsymbol{J}\boldsymbol{\Delta} + \boldsymbol{\Delta}^T \boldsymbol{J}^T \boldsymbol{J}\boldsymbol{\Delta} \end{aligned}$$

One term is the transpose of the other, but since both result in a scalar transposition doesn't matter.

where $\boldsymbol{J}^T\boldsymbol{J} \in \mathbb{R}^{n\times n}$ is the *approximate* Hessian from page 618.

To minimize the error of this linearized least squares system we take the derivative with respect to $\boldsymbol{\Delta}$ and set it to zero

$$\frac{\mathrm{d}F}{\mathrm{d}\boldsymbol{\Delta}} = 0 \Rightarrow 2\boldsymbol{f}_0^T \boldsymbol{J} + 2\boldsymbol{J}^T\boldsymbol{J}\boldsymbol{\Delta} = 0$$

which we can solve for the locally optimal update

$$\boldsymbol{\Delta} = -\left(\boldsymbol{J}^T\boldsymbol{J}\right)^{-1} \boldsymbol{J}^T \boldsymbol{f}_0 \tag{F.3}$$

where we can recognize the pseudo or left generalized-inverse of $\boldsymbol{J}$. Once again we iterate to find the solution – a Gauss-Newton iteration.

Numerical Issues

When solving Eq. F.3 we may find that the Hessian $\boldsymbol{J}^T\boldsymbol{J}$ is poorly conditioned or singular and this can be remedied by adding a damping term

$$\boldsymbol{\Delta} = -\left(\boldsymbol{J}^T\boldsymbol{J} + \lambda\boldsymbol{I}\right)^{-1} \boldsymbol{J}^T \boldsymbol{f}_0$$

which makes the system more positive definite. Since $\boldsymbol{J}^T\boldsymbol{J} + \lambda\boldsymbol{I}$ is effectively in the denominator, increasing λ will decrease $\|\boldsymbol{\Delta}\|$ and slow convergence.

How do we choose λ? We can experiment with different values but a better way is the Levenberg-Marquardt algorithm (Algorithm F.1) which adjusts λ to ensure convergence. If the error increases compared to the last step then the step is repeated with increased λ to reduce the step size. If the error decreases then λ is reduced to increase the convergence rate. The updates vary continuously between Gauss-Newton (low λ) and gradient descent (high λ).

For problems where n is large inverting the $n \times n$ approximate Hessian is expensive. Typically $m < n$ which means the Jacobian is not square and Eq. F.3 can be rewritten as

```
initialize $\lambda$
repeat
    compute $\boldsymbol{f}_0 = \boldsymbol{f}(\boldsymbol{x}_0), \boldsymbol{J} = \boldsymbol{J}(\boldsymbol{x}_0), \boldsymbol{H} = \boldsymbol{J}^T\boldsymbol{J}$
    $\Delta = -(\boldsymbol{H} + \lambda \boldsymbol{I})^{-1}\boldsymbol{J}^T\boldsymbol{f}_0$
    if $F(\boldsymbol{x}_0 + \Delta) < F(\boldsymbol{x}_0)$ then
        – error decreased: reduce damping
        $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}_0 + \Delta$
        $\lambda \leftarrow \lambda / c$
    else
        – error increased: discard and raise damping
        $\lambda \leftarrow c\lambda$
    end
until $\|\boldsymbol{\Delta}\| < \varepsilon$
```

Algorithm F.1.
Levenberg-Marquardt algorithm, *c* is typically chosen in the range 2 to 10

$$\Delta = -\boldsymbol{J}^T\left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1}\boldsymbol{f}_0$$

which is the right pseudo-inverse and involves inverting a smaller matrix. We can reintroduce a damping term

$$\Delta = -\boldsymbol{J}^T\left(\boldsymbol{J}\boldsymbol{J}^T + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{f}_0$$

and if λ is large this becomes simply

$$\Delta \approx -\beta\boldsymbol{J}^T\boldsymbol{f}_0$$

but exhibits very slow convergence.

If $\boldsymbol{f}_k(\cdot)$ has additive noise that is zero mean, normally distributed and time invariant we have a maximum likelihood estimator of $\boldsymbol{x}$. Outlier data has a significant impact on the result since errors are squared. Robust estimators minimize the effect of outlier data and in an M-estimator

$$F(\boldsymbol{x}) = \rho\left(\boldsymbol{f}_k(\boldsymbol{x})\right)$$

the squared norm is replaced by a loss function $\rho(\cdot)$ which models the likelihood of its argument. Unlike the squared norm these functions flatten off for large values, and some common examples include the Huber loss function and the Tukey biweight function.

F.2.4 Sparse Nonlinear Least Squares

For a large class of problems the overall cost is the sum of quadratic costs

$$F(\boldsymbol{x}) = \sum_k \left\|\boldsymbol{f}_k(\boldsymbol{x})\right\|^2 = \sum_k \boldsymbol{f}_k(\boldsymbol{x})^T\boldsymbol{f}_k(\boldsymbol{x}) \tag{F.4}$$

Consider the problem of fitting a model $\boldsymbol{z} = \phi(\boldsymbol{w}; \boldsymbol{x})$ where $\phi: \mathbb{R}^p \mapsto \mathbb{R}^m$ with parameters $\boldsymbol{x} \in \mathbb{R}^n$ to a set of data points $(\boldsymbol{w}_k, \boldsymbol{z}_k)$. The error vector associated with the kth data point is

$$\boldsymbol{f}_k(\boldsymbol{x}) = \boldsymbol{z}_k - \phi\left(\boldsymbol{w}_k; \boldsymbol{x}\right) \in \mathbb{R}^m$$

and minimizing Eq. F.4 gives the optimal model parameters $\boldsymbol{x}$.

Another example is pose-graph optimization as used for pose-graph SLAM and bundle adjustment. Edge k in the graph connects vertices i and j and has an associated cost $\boldsymbol{f}_k(\cdot): \mathbb{R}^n \mapsto \mathbb{R}^m$

$$\boldsymbol{f}_k(\boldsymbol{x}) = \hat{\boldsymbol{e}}_k(\boldsymbol{x}) - \boldsymbol{e}_k^\# \tag{F.5}$$

where $\boldsymbol{e}_k^\#$ is the observed value of the edge parameter and $\hat{\boldsymbol{e}}_k(\boldsymbol{x})$ is the estimate based on the state $\boldsymbol{x}$ of the pose graph. This is linearized

$$\boldsymbol{f}_k'(\boldsymbol{\Delta}) \approx \boldsymbol{f}_{0,k} + \boldsymbol{J}_k \boldsymbol{\Delta}$$

and the squared error for the edge is

$$F_k(\boldsymbol{x}) = \boldsymbol{f}_k(\boldsymbol{x})^T \boldsymbol{\Omega}_k \boldsymbol{f}_k(\boldsymbol{x})$$

where $\boldsymbol{\Omega}_k \in \mathbb{R}^{m \times m}$ is a positive-definite constant matrix which we combine as

This can be used to specify the significance of the edge det$\boldsymbol{\Omega}_k$ with respect to other edges, as well as the relative significance of the elements of $\boldsymbol{f}_k(\cdot)$.

$$\begin{aligned} F_k(\boldsymbol{\Delta}) &\approx \left(\boldsymbol{f}_{0,k} + \boldsymbol{J}_k\boldsymbol{\Delta}\right)^T \boldsymbol{\Omega}_k \left(\boldsymbol{f}_{0,k} + \boldsymbol{J}_k\boldsymbol{\Delta}\right) \\ &\approx \boldsymbol{f}_{0,k}^T \boldsymbol{\Omega}_k \boldsymbol{f}_{0,k} + \boldsymbol{f}_{0,k}^T \boldsymbol{\Omega}_k \boldsymbol{J}_k \boldsymbol{\Delta} + \boldsymbol{\Delta}^T \boldsymbol{J}_k^T \boldsymbol{\Omega}_k \boldsymbol{f}_{0,k} + \boldsymbol{\Delta}^T \boldsymbol{J}_k^T \boldsymbol{\Omega}_k \boldsymbol{J}_k \boldsymbol{\Delta} \\ &\approx c_k + 2\boldsymbol{b}_k^T \boldsymbol{\Delta} + \boldsymbol{\Delta}^T \boldsymbol{H}_k \boldsymbol{\Delta} \end{aligned}$$

where $\boldsymbol{b}_k^T = \boldsymbol{f}_{0,k}^T \boldsymbol{\Omega}_k \boldsymbol{J}_k$ and $\boldsymbol{H}_k = \Sigma_k \boldsymbol{J}_k^T \boldsymbol{\Omega}_k \boldsymbol{J}_k$. The total cost is the sum of all edge costs

$$\begin{aligned} F(\boldsymbol{\Delta}) &= \sum_k F_k(\boldsymbol{\Delta}) \\ &\approx \sum_k \left(c_k + 2\boldsymbol{b}_k^T \boldsymbol{\Delta} + \boldsymbol{\Delta}^T \boldsymbol{H}_k \boldsymbol{\Delta}\right) \\ &\approx \sum_k c_k + 2\left(\sum_k \boldsymbol{b}_k^T\right)\boldsymbol{\Delta} + \boldsymbol{\Delta}^T \left(\sum_k \boldsymbol{H}_k\right)\boldsymbol{\Delta} \\ &\approx c + 2\boldsymbol{b}^T \boldsymbol{\Delta} + \boldsymbol{\Delta}^T \boldsymbol{H} \boldsymbol{\Delta} \end{aligned}$$

where $\boldsymbol{b}^T = \Sigma_k \boldsymbol{f}_{0,k}^T \boldsymbol{\Omega}_k \boldsymbol{J}_k$ and $\boldsymbol{H} = \Sigma_k \boldsymbol{J}_k^T \boldsymbol{\Omega}_k \boldsymbol{J}_k$ are summations over the edges of the graph. Once they are computed we proceed as previously, taking the derivative with respect to $\boldsymbol{\Delta}$ and setting it to zero, solving for the update $\boldsymbol{\Delta}$ and iterating using Algorithm F.1.

State Vector

The state vector is a concatenation of all poses and coordinates in the optimization problem. For pose-graph SLAM it takes the form

$$\boldsymbol{x} = \left\{\xi_1, \xi_2 \cdots \xi_N\right\} \in \mathbb{R}^n$$

Poses must be represented in a vector form and preferably one that is compact and singularity free. For **SE**(2) this is quite straightforward and we use $\xi \sim (x, y, \theta) \in \mathbb{R}^3$. For **SE**(3) we will use $\xi \sim (\boldsymbol{t}, \boldsymbol{r}) \in \mathbb{R}^6$ which comprises translation $\boldsymbol{t} \in \mathbb{R}^3$ and rotation $\boldsymbol{r} \in \mathbb{R}^3$. The latter can be triple angles (Euler or roll-pitch-yaw), axis-angle, exponential coordinates or the vector part of a unit-quaternion as discussed on page 499. The state vector has structure, comprising a sequence of subvectors one per pose. We denote the i^{th} subvector of $\boldsymbol{x}$ as $\boldsymbol{x}_i \in \mathbb{R}^{N_\xi}$, where $N_\xi = 3$ for **SE**(2) and $N_\xi = 6$ for **SE**(3).

For pose-graph SLAM with landmarks, or bundle adjustment the state vector comprises poses and coordinate vectors

$$\boldsymbol{x} = \left\{\xi_1, \xi_2 \cdots \xi_N \,|\, \boldsymbol{P}_1, \boldsymbol{P}_2 \cdots \boldsymbol{P}_M\right\} \in \mathbb{R}^n$$

and the i^{th} and j^{th} subvectors of $\boldsymbol{x}$ are denoted $\boldsymbol{x}_i \in \mathbb{R}^{N_\xi}$ and $\boldsymbol{x}_j \in \mathbb{R}^{N_P}$ and correspond to ξ_i and $\boldsymbol{P}_j$ respectively.

Inherent Structure

A key observation is that the error vector $\boldsymbol{f}_k(\boldsymbol{x})$ for edge k depends only on the associated vertices i and j, and this means that the Jacobian

$$\boldsymbol{J}_k = \frac{\partial \boldsymbol{f}_k(\boldsymbol{x})}{\partial \boldsymbol{x}} \in \mathbb{R}^{m \times m}$$

is mostly zeros

$$\boldsymbol{J}_k = \left(0 \cdots \boldsymbol{A}_i \cdots \boldsymbol{B}_j \cdots 0\right),\ \boldsymbol{A}_i = \frac{\partial \boldsymbol{f}_k(\boldsymbol{x})}{\partial \boldsymbol{x}_i},\ \boldsymbol{B}_j = \frac{\partial \boldsymbol{f}_k(\boldsymbol{x})}{\partial \boldsymbol{x}_j}$$

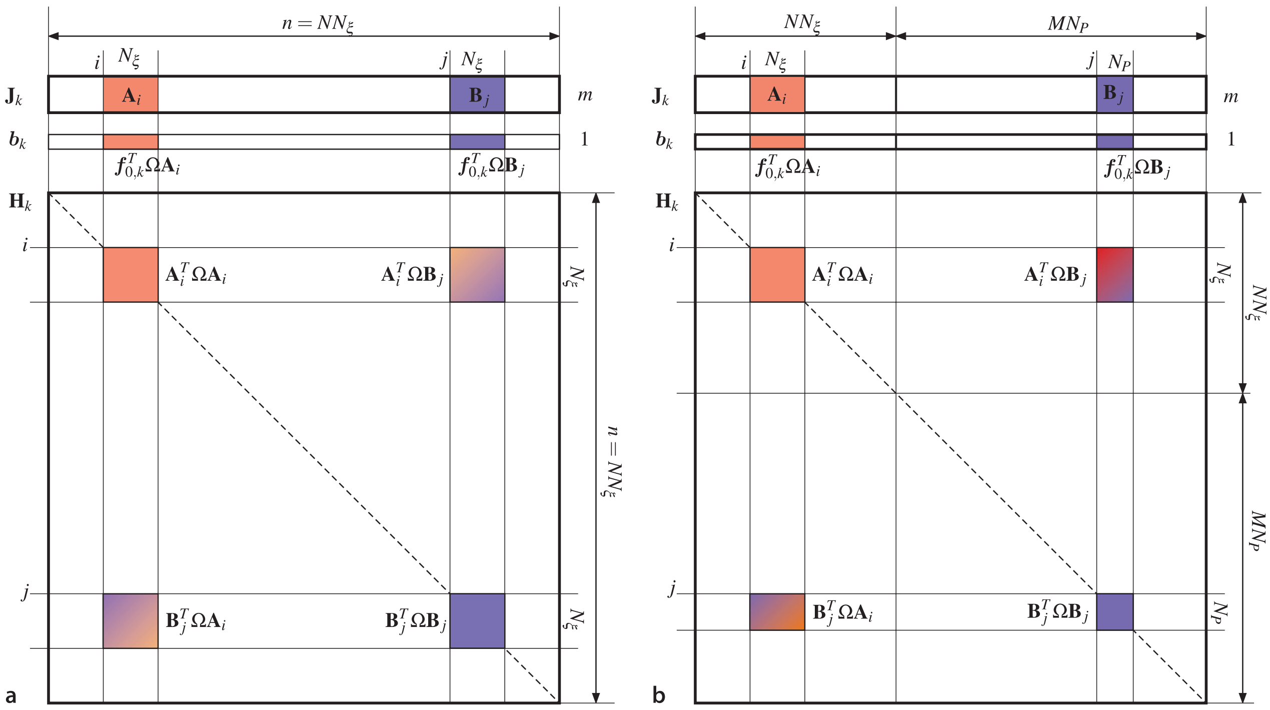
where $\boldsymbol{A}_i \in \mathbb{R}^{m \times N_\xi}$ and $\boldsymbol{B}_j \in \mathbb{R}^{m \times N_\xi}$ or $\boldsymbol{B}_j \in \mathbb{R}^{m \times N_P}$ according to the state vector structure.

This sparse block structure means that the vector $\boldsymbol{b}_k$ and the Hessian $\boldsymbol{J}_k^T \boldsymbol{\Omega}_k \boldsymbol{J}_k$ also have a sparse block structure as shown in Fig. F.1. The Hessian has just four small nonzero blocks so rather than compute the product $\boldsymbol{J}_k^T \boldsymbol{\Omega}_k \boldsymbol{J}_k$, which involves many multiplications by zero, we can just compute the four nonzero blocks and add them into the Hessian for the least squares system. All blocks in a row have the same height, and in a column have the same width. For pose-graph SLAM with landmarks, or bundle adjustment the blocks are of different sizes as shown in Fig. F.1b.

If the value of an edge represents pose then Eq. F.5 must be replaced with $\boldsymbol{f}_k(\boldsymbol{x}) = \hat{\boldsymbol{e}}_k(\boldsymbol{x}) \ominus \boldsymbol{e}_k^{\#}$. We generalize this with the $\boxminus$ operator to indicate that the use of $-$ or $\ominus$ as appropriate. Similarly when updating the state vector at the end of an iteration the poses must be compounded $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}_0 \oplus \boldsymbol{\Delta}$ and we generalize this to the $\boxplus$ operator. The pose-graph optimization is solved by the iteration in Algorithm F.2.

```
repeat
    $H \leftarrow 0,\ b \leftarrow 0$
    for each $k$ do
        $f_{0,k}(x_0) = \hat{e}_k(x) \boxminus e_k^{\#}$
        $(i, j) = \text{vertices}(k)$
        compute $A_i(x_i),\ B_j(x_j)$
        $b_i \leftarrow b_i + f_{0,k}^T \Omega_k A_i$
        $b_j \leftarrow b_j + f_{0,k}^T \Omega_k B_j$
        $H_{i,i} \leftarrow H_{i,i} + A_i^T \Omega_k A_i$
        $H_{i,j} \leftarrow H_{i,j} + A_i^T \Omega_k B_j$
        $H_{j,i} \leftarrow H_{j,i} + B_j^T \Omega_k A_i$
        $H_{j,j} \leftarrow H_{j,j} + B_j^T \Omega_k B_j$
    end
    $\Delta = -H^{-1} b$
    $x_0 \leftarrow x_0 \boxplus \Delta$
until $\|\Delta\| < \varepsilon$
```

Alogorithm F.2.
Pose graph optimization. For Levenberg-Marquardt optimization replace line 14 with lines 4–12 from Algorithm F.1



$n = NN_\xi$
i N_ξ j N_ξ
$\mathbf{J}_k$ $\mathbf{A}_i$ $\mathbf{B}_j$ m
$\boldsymbol{b}_k$ 1
$\boldsymbol{f}_{0,k}^T\Omega\mathbf{A}_i$ $\boldsymbol{f}_{0,k}^T\Omega\mathbf{B}_j$
$\mathbf{H}_k$
$\mathbf{A}_i^T\Omega\mathbf{A}_i$ $\mathbf{A}_i^T\Omega\mathbf{B}_j$
$\mathbf{B}_j^T\Omega\mathbf{A}_i$ $\mathbf{B}_j^T\Omega\mathbf{B}_j$
$n = NN_\xi$
a
NN_ξ MN_P
i N_ξ j N_P
N_ξ N_P
b

Fig. F.1. Inherent structure of the error vector, Jacobian and Hessian matrices for graph-based least-squares problems. **a** Pose-graph SLAM with N nodes representing robot pose as $\mathbb{R}^{N_\xi}$; **b** bundle adjustment with N nodes representing camera pose as $\mathbb{R}^{N_\xi}$ and M nodes representing landmark position as $\mathbb{R}^{N_P}$. The indices i and j denote the ith and jth block not the ith and jth row or column. *White* indicates zero values

Large Scale Problems

For pose-graph SLAM with thousands of poses or bundle adjustment with thousands of cameras and millions of landmarks the Hessian matrix will be massive leading to computation and storage challenges. The overall Hessian is the summation of many edge Hessians structured as shown in Fig. F.1 and the total Hessian for two problems we have discussed are shown in Fig. F.2. They have clear structure which we can exploit.

Firstly, in both cases the Hessian is sparse – that is, it contains mostly zeros. MATLAB has built-in support for such matrices and instead of storing all those zeros (at 8 bytes each) it simply keeps a list of the nonzero elements. All the standard matrix operations employ efficient algorithms for manipulating sparse matrices.

Secondly, for the bundle adjustment case we see that the Hessian has two block diagonal submatrices so we partition the system as

$$\begin{pmatrix} \boldsymbol{B} & \boldsymbol{E} \\ \boldsymbol{E}^T & \boldsymbol{C} \end{pmatrix} \begin{pmatrix} \Delta_\xi \\ \Delta_P \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_\xi \\ \boldsymbol{b}_P \end{pmatrix}$$

where $\boldsymbol{B}$ and $\boldsymbol{C}$ are block diagonal.▸ The subscripts ξ and P denote the blocks of $\boldsymbol{\Delta}$ and $\boldsymbol{b}$ associated with camera poses and landmark positions respectively. We solve first for the camera pose updates $\boldsymbol{\Delta}_\xi$

A block diagonal matrix is inverted by simply inverting each of the nonzero blocks along the diagonal.

$$\boldsymbol{S}\boldsymbol{\Delta}_\xi = \boldsymbol{b}_\xi - \boldsymbol{E}\boldsymbol{C}^{-1}\boldsymbol{b}_P$$

where $\boldsymbol{S} = \boldsymbol{B} - \boldsymbol{E}\boldsymbol{C}^{-1}\boldsymbol{E}^T$ is the Schur complement which is a symmetric positive-definite matrix that is also block diagonal. Then we solve for the update to landmark positions

$$\boldsymbol{\Delta}_P = \boldsymbol{C}^{-1}\left(\boldsymbol{b}_P + \boldsymbol{E}^T\boldsymbol{\Delta}_\xi\right)$$

More sophisticated techniques exploit the fine-scale block structure to further reduce computational time, for example GTSAM (https://bitbucket.org/gtborg/gtsam) and SLAM++ (https://sourceforge.net/projects/slam-plus-plus).

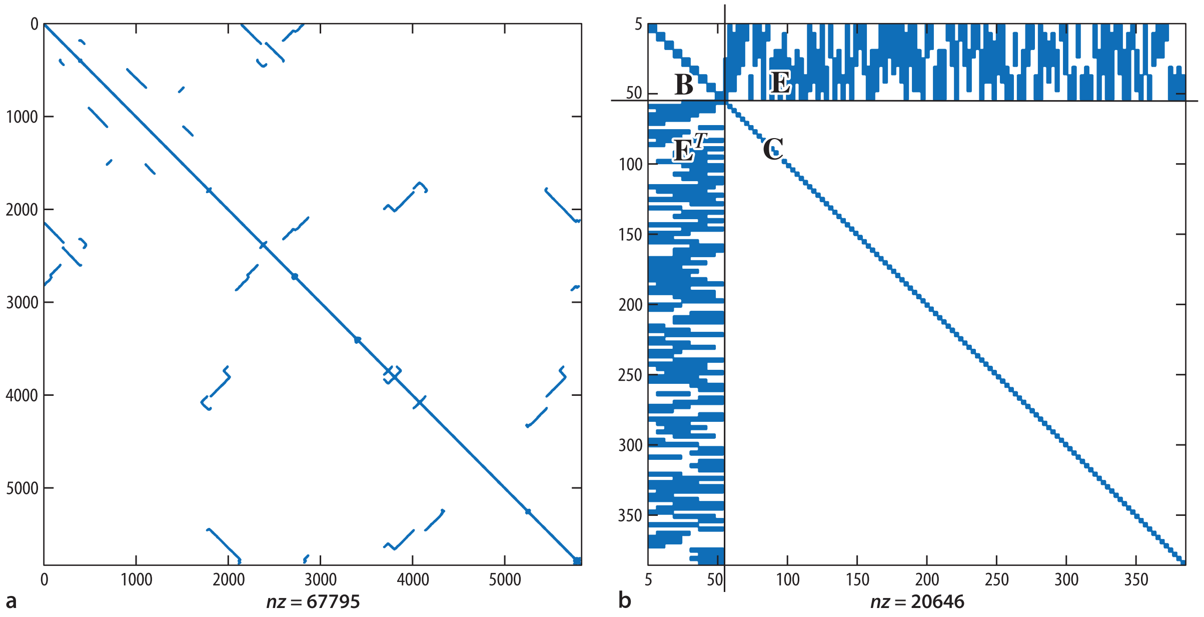


Fig. F.2. Hessian sparsity maps produced using the MATLAB `spy` function, the number of non-zero elements is shown beneath the plot. **a** Hessian for the pose-graph SLAM problem of Fig. 6.17, the diagonal elements represent pose constraints between successive nodes due to odometry, the off-diagonal terms represent constraints due to revisiting locations (loop closures); **b** Hessian for a bundle adjustment problem with 10 cameras and 110 landmarks (`vision/examples/bademo.m`)

Anchoring

Optimization provides a solution where the *relative* poses and positions give the lowest overall cost, and the solution will have an arbitrary transformation with respect to a global reference frame. To obtain absolute poses and positions we must anchor or fix some nodes – assign them values with respect to the global frame and prevent the optimization from adjusting them. The appropriate way to achieve this is to remove from $\boldsymbol{H}$ and $\boldsymbol{b}$ the rows and columns corresponding to the anchored poses and positions. We then solve a lower dimensional problem for $\boldsymbol{\Delta}'$ which will be shorter than $\boldsymbol{x}$ and careful book keeping is required to correctly match the subvectors of $\boldsymbol{\Delta}'$ with those of $\boldsymbol{x}$ for the update.

G Gaussian Random Variables

The 1-dimensional Gaussian function

$$g(x) = \frac{1}{\sqrt{\sigma^2 2\pi}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \tag{G.1}$$

is described by the position of its peak μ and its width σ. The total area under the curve is unity and $g(x) > 0, \forall x$. The function can be plotted using the Toolbox function `gaussfunc`

```
>> x = linspace(-6, 6, 500);
>> plot(x, gaussfunc(0, 1, x), 'r' )
>> hold on
>> plot(x, gaussfunc(0, 2^2, x), '--b' )
```

and Fig. G.1 shows two Gaussians with zero mean and $\sigma = 1$ and $\sigma = 2$. Note that the second argument to `gaussfunc` is the variance not standard deviation.

If the Gaussian is considered to be a probability density function (PDF) then this is the well known normal distribution and the peak position μ is the mean value and the width σ is the standard deviation. A random variable drawn from a normal distribution is often written as $X \sim N(\mu, \sigma^2)$, and $N(0, 1)$ is referred to as the standard normal distribution – the MATLAB function `randn` draws random numbers from this distribution. To draw one hundred Gaussian random numbers with mean `mu` and standard deviation `sigma` is

```
>> g = sigma * randn(100) + mu;
```

The probability that a random value falls within an interval $x \in [x_1, x_2]$ is obtained by integration

$$P = \int_{x_1}^{x_2} g(x)\mathrm{d}x = \Phi(x_2) - \Phi(x_1) \quad \text{where} \quad \Phi(x) = \int_{-\infty}^{x} g(x)\mathrm{d}x$$

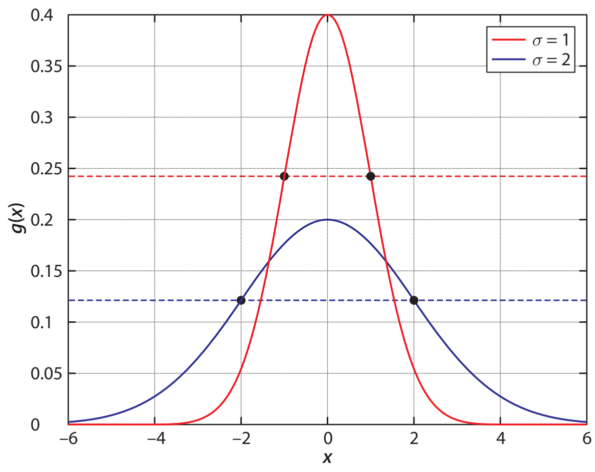


Fig. G.1. Two Gaussian functions, both with with mean $\mu = 0$, and with standard deviation $\sigma = 1$, and $\sigma = 2$. The markers indicate the points $x = \mu \pm 1\sigma$. The *blue curve* is wider but less tall, since the total area under the curve is unity

or evaluation of the cumulative normal distribution function $\Phi(x)$. The marked points in Fig. G.1 at $\mu \pm 1\sigma$ delimit the 1σ confidence interval. The area under the curve over this interval is 0.68, so the probability of a random value being drawn from this interval is 68%.

The Gaussian can be extended to an arbitrary number of dimensions. The n-dimensional Gaussian, or multivariate normal distribution, is

$$g(\boldsymbol{x}) = \frac{1}{\sqrt{\det(\boldsymbol{P})(2\pi)^n}} \mathrm{e}^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{P}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{G.2}$$

and compared to the scalar case of Eq. G.1 $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{\mu} \in \mathbb{R}^n$ have become vectors, the squared term in the exponent has been replaced by a matrix quadratic form, and σ^2, the variance, has become a positive-definite (and hence symmetric) covariance matrix $\boldsymbol{P} \in \mathbb{R}^{n \times n}$. The diagonal elements represent the variance of x_i and the off-diagonal elements $\boldsymbol{P}_{ij}$ are the correlationss between x_i and x_j. If the variables are independent or uncorrelated the matrix $\boldsymbol{P}$ would be diagonal. The covariance matrix is symmetric and positive definite.

We can plot a 2-dimensional Gaussian

```
>> [x,y] = meshgrid(-5:0.1:5, -5:0.1:5);
>> P = diag([1 2^2]);
>> surfc(x, y, gaussfunc([0 0], P, x, y))
```

as a surface which is shown in Fig. G.2. In this case $\boldsymbol{\mu} = (0, 0)$ and $\boldsymbol{P} = \mathrm{diag}(1^2, 2^2)$ which corresponds to uncorrelated variables with standard deviation of 1 and 2 respectively. Figure G.2 also shows a number of elliptical contours – contours of constant probability density. If this 2-dimensional probability density function represents the position of a robot in the xy-plane the most likely position for the robot is at $(0, 0)$ and the size of the ellipse says something about our spatial certainty. A particular contour indicates the boundary of a region within which the robot is located with a particular probability. A large ellipse indicates we know, with that probability, that the robot is somewhere inside a large area – we have low certainty about the robot's position. Conversely, a small ellipse means that we know the robot, with the same probability, is somewhere within a much smaller area.

The contour lines are ellipses and in this example the radii in the y- and x-directions are in the ratio 2 : 1 as defined by the ratio of the standard deviations. For higher order Gaussians, $n > 2$, the corresponding confidence interval is the surface of an ellipsoid in n-dimensional space.

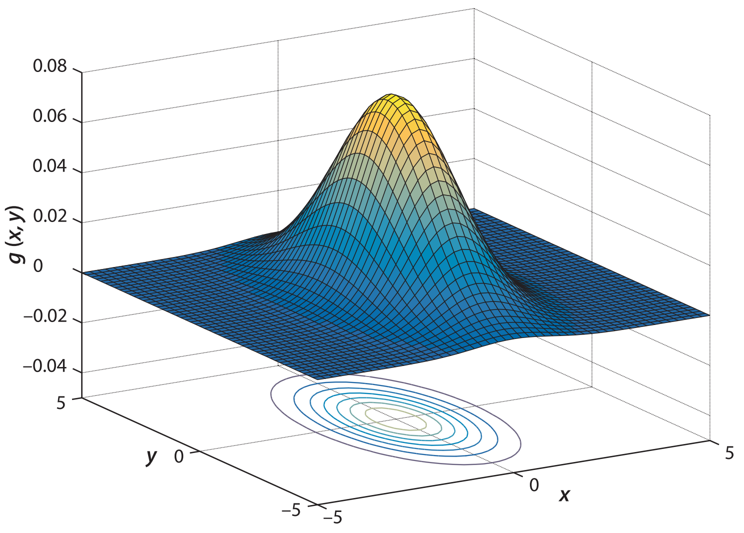


Fig. G.2. The 2-dimensional Gaussian with covariance $\boldsymbol{P} = \mathrm{diag}(1^2, 2^2)$. Contours lines of constant probability density are shown beneath

The connection between Gaussian probability density functions and ellipses can be found in the quadratic exponent of Eq. G.2 which is the equation of an ellipse or ellipsoid. All the points that satisfy

It is also the definition of Mahalanobis distance, the covariance weighted distance between $\boldsymbol{x}$ and $\boldsymbol{\mu}$.

$$(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{P}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = s$$

result in a constant probability density value, that is, a contour of the 2-dimensional Gaussian. s is related to the probability by

$$s = \chi^2_n(p)$$

which is the χ^2 distribution with n degrees of freedom, 2 in this case, and p is the probability that the point $\boldsymbol{x}$ lies on the ellipse. For example the 50% confidence interval is

If we draw a vector of length n from the multivariate Gaussian each element is normally distributed. The sum of squares of independent normally distributed values is known to be distributed according to a χ^2 (chi-squared) distribution with n degrees of freedom.

```
>> s = chi2inv(0.5, 2)
s =
    1.3863
```

where the first argument is the probability and the second is the number of degrees of freedom.

This function requires the MATLAB Statistics and Machine Learning Toolbox™. The Robotics Toolbox provides `chi2inv_rtb` which is an approximation for the case $n=2$.

If the covariance matrix is diagonal then the ellipse is aligned with the x- and y-axes as we saw in Sect. C.1.4. This indicates that the two variables are independent and have zero correlation. Conversely a rotated ellipse indicates that the covariance is not diagonal and the two variables are correlated.

To draw a covariance ellipse we use the general approach for ellipses outlined in Sect. C.1.4 but the right-hand side of the ellipse equation is s not 1, and $\boldsymbol{E} \equiv \boldsymbol{P}$.

H Kalman Filter

Consider the system shown in Fig. H.1. The physical robot is a "black box" which has a true state or pose $\boldsymbol{x}$ that evolves over time according to the applied inputs. We cannot directly measure the state, but sensors on the robot have outputs which are a function of that true state. Our challenge is: given the system inputs and sensor outputs *estimate* the unknown true state $\boldsymbol{x}$ and how certain we are of that estimate.

At face value this might seem hard, or even impossible, but there are quite a lot of things we know about system that will help us. Firstly, we know how the state evolves over time as a function of the inputs – this is the state transition model $\boldsymbol{f}(\cdot)$, and we know the inputs to the system $\boldsymbol{u}$. Our model is unlikely to be perfect and it is common to represent this uncertainty by an imaginary random number generator which is corrupting the system state – process noise. Secondly, we know how the sensor output depends on the state – this is the sensor model $\boldsymbol{h}(\cdot)$ and its uncertainty is also modeled by an imaginary random number generator – sensor noise.

Often called the process or motion model.

For example wheel slippage on a mobile ground robot or wind gusts for a UAV.

The imaginary random number sources $\boldsymbol{v}$ and $\boldsymbol{w}$ are *inside* the black box so the random numbers are also unknowable. However we can describe the characteristics of these random numbers – their *distribution* which tells us how likely it is that we will *draw* a random number with a particular value. A lot of noise in physical systems can be modeled well by the Gaussian (aka normal) distribution $\mathcal{N}(\mu, \sigma^2)$ which is characterized by a mean μ and a standard deviation σ. There are infinitely many possible distributions but the Gaussian distribution has some nice mathematical properties that we will rely on. However we should never assume that noise is Gaussian – we should attempt to determine the distribution by understanding the physics of the process and the sensor, or from careful measurement and analysis.

Which can be nonsymmetrical or have multiple peaks.

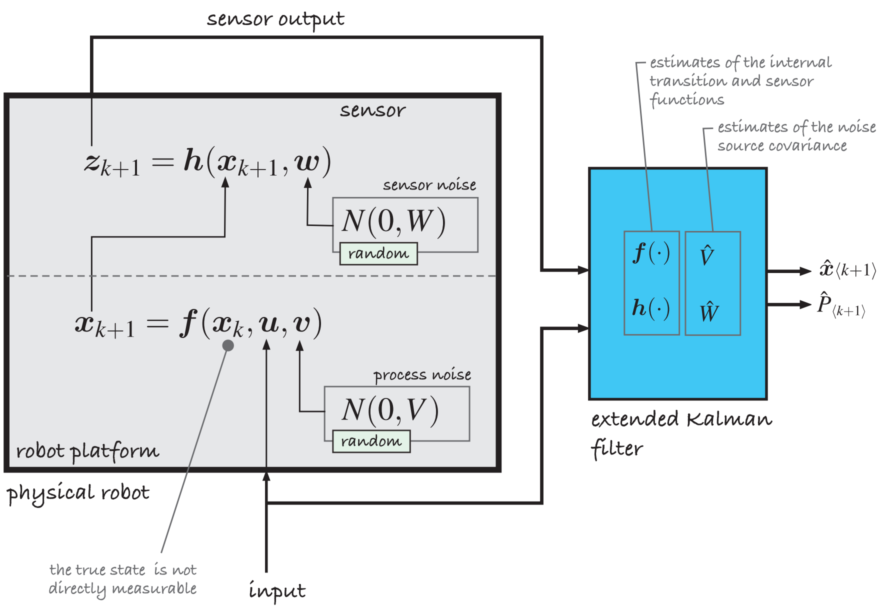

sensor output
sensor
$z_{k+1} = h(x_{k+1}, w)$
sensor noise
$N(0, W)$
random
$x_{k+1} = f(x_k, u, v)$
process noise
$N(0, V)$
random
robot platform
physical robot
the true state is not directly measurable
input
estimates of the internal transition and sensor functions
estimates of the noise source covariance
$f(\cdot)$
$h(\cdot)$
$\hat{V}$
$\hat{W}$
$\hat{x}\langle k+1\rangle$
$\hat{P}_{\langle k+1\rangle}$
extended Kalman filter

Fig. H.1. The physical robot on the left has a true state that cannot be directly measured, however we gain a clue from the sensor output which is a function of this unknown true state

In general terms, the problem we wish to solve is:

given a model of the system $\boldsymbol{f}(\cdot)$, $\boldsymbol{h}(\cdot)$, $\hat{\boldsymbol{V}}$ and $\hat{\boldsymbol{W}}$; the known inputs applied to the system $\boldsymbol{u}$; and some noisy sensor measurements $\boldsymbol{z}$, find an estimate $\hat{\boldsymbol{x}}$ of the system state and our uncertainty $\hat{\boldsymbol{P}}$ in that estimate.

In a robotic localization context $\boldsymbol{x}$ is the unknown position or pose of the robot, $\boldsymbol{u}$ is the commands sent to the motors and $\boldsymbol{z}$ is the output of various sensors on the robot. For a ground robot $\boldsymbol{x}$ would be the pose in $\mathbf{SE}(2)$ and $\boldsymbol{u}$ would be the motor commands and $\boldsymbol{z}$ might be the measured odometry or range and bearing to landmarks. For a flying robot $\boldsymbol{x}$ would be the pose in $\mathbf{SE}(3)$ and $\boldsymbol{u}$ are the known forces applied to the airframe and $\boldsymbol{z}$ might be the measured accelerations and angular velocities.▸

The state is a vector and there are many approaches to mapping pose to a vector, especially the rotational component – Euler angles, quaternions, and exponential coordinates are commonly used.

H.1 Linear Systems – Kalman Filter

Consider the transition model described as a discrete-time linear time-invariant system

$$\boldsymbol{x}\langle k{+}1\rangle = \boldsymbol{F}\boldsymbol{x}\langle k\rangle + \boldsymbol{G}\boldsymbol{u}\langle k\rangle + \boldsymbol{v}\langle k\rangle \tag{H.1}$$

$$\boldsymbol{z}\langle k\rangle = \boldsymbol{H}\boldsymbol{x}\langle k\rangle + \boldsymbol{w}\langle k\rangle \tag{H.2}$$

where k is the time step, $\boldsymbol{x} \in \mathbb{R}^n$ is the state vector, and $\boldsymbol{u} \in \mathbb{R}^m$ is a vector of inputs to the system at time k, for example a velocity command, or applied forces and torques. The matrix $\boldsymbol{F} \in \mathbb{R}^{n\times n}$ describes the dynamics of the system, that is, how the states evolve with time. The matrix $\boldsymbol{G} \in \mathbb{R}^{n\times m}$ describes how the inputs are coupled to the system states. The vector $\boldsymbol{z} \in \mathbb{R}^p$ represents the outputs of the system as measured by sensors. The matrix $\boldsymbol{H} \in \mathbb{R}^{p\times n}$ describes how the system states are mapped to the system outputs which we can observe.

To account for errors in the motion model ($\boldsymbol{F}$ and $\boldsymbol{G}$) or unmodeled disturbances we introduce a Gaussian random variable $\boldsymbol{v} \in \mathbb{R}^n$ termed the process noise. $\boldsymbol{v}\langle k\rangle \sim N(0, \boldsymbol{V})$, that is, it has zero mean and covariance $\boldsymbol{V} \in \mathbb{R}^{n\times n}$. Covariance is a matrix quantity which is the variance for a multi-dimensional distribution – it is a positive definite matrix and therefore symmetric. The sensor measurement model $\boldsymbol{H}$ is not perfect either and this is modeled by sensor measurement noise, a Gaussian random variable $\boldsymbol{w} \in \mathbb{R}^p$, $\boldsymbol{w}\langle k\rangle \sim N(0, \boldsymbol{W})$ and covariance $\boldsymbol{W} \in \mathbb{R}^{p\times p}$.

The Kalman filter is an optimal estimator for the case where the process and measurement noise are zero-mean Gaussian noise. The filter has two steps: prediction and update. The prediction is based on the previous state and the inputs that were applied

$$\hat{\boldsymbol{x}}^{+}\langle k{+}1\rangle = \boldsymbol{F}\hat{\boldsymbol{x}}\langle k\rangle + \boldsymbol{G}\boldsymbol{u}\langle k\rangle \tag{H.3}$$

$$\hat{\boldsymbol{P}}^{+}\langle k{+}1\rangle = \underbrace{\boldsymbol{F}\hat{\boldsymbol{P}}\langle k\rangle\boldsymbol{F}^T}_{} + \hat{\boldsymbol{V}} \tag{H.4}$$

where $\hat{\boldsymbol{x}}$ is the estimate of the state and $\hat{\boldsymbol{P}} \in \mathbb{R}^{n\times n}$ is the estimated covariance, or uncertainty, in $\hat{\boldsymbol{x}}$. The notation $^+$ makes explicit that the left-hand side is an estimate at time $k+1$ based on information from time k. $\hat{\boldsymbol{V}}$ is our best estimate of the covariance of the process noise.

The indicated term in Eq. H.4 *projects* the estimated covariance from the current time step to the next. Consider a one dimensional example where F is a scalar and the state estimate $\hat{x}\langle k\rangle$ has a PDF which is Gaussian with a mean $\bar{x}\langle k\rangle$ and a variance $\sigma^2\langle k\rangle$. The prediction equation maps the state and its Gaussian distribution to a new Gaussian distribution with a mean $F\bar{x}\langle k\rangle$ and a variance $F^2\sigma^2\langle k\rangle$. The term $\boldsymbol{F}\boldsymbol{P}\langle k\rangle\boldsymbol{F}\langle k\rangle^T$ is the matrix form of this since

$$\operatorname{cov}(\boldsymbol{F}\boldsymbol{x}) = \boldsymbol{F}\operatorname{cov}(\boldsymbol{x})\boldsymbol{F}^T \tag{H.5}$$

which scales the covariance appropriately.

The prediction of $\hat{\boldsymbol{P}}$ involves the addition of two positive-definite matrices so the uncertainty will increase – this is to be expected since we have used an uncertain model to predict the future value of an already uncertain estimate. $\hat{\boldsymbol{V}}$ must be a reasonable estimate of the covariance of the actual process noise. If we overestimate it, that is our estimate of process noise is larger than it really is, then we will have a large increase in uncertainty at this step, a pessimistic estimate of our certainty.

To counter this growth in uncertainty we need to introduce new information such as measurements made by the sensors since they depend on the state. The difference between what the sensors measure and what the sensors are predicted to measure is

$$\nu = \boldsymbol{z}^{\#}\langle k+1\rangle - \boldsymbol{H}\hat{\boldsymbol{x}}^{+}\langle k+1\rangle \in \mathbb{R}^{p}$$

Some of this difference is due to noise in the sensor, the measurement noise, but the remainder provides valuable information related to the error between the actual and the predicted value of the state. Rather than considering this as error we refer to it more positively as *innovation* – new information.

The second step of the Kalman filter, the *update* step, maps the innovation into a correction for the predicted state, optimally tweaking the estimate based on what the sensors observed

$$\hat{\boldsymbol{x}}\langle k+1\rangle = \hat{\boldsymbol{x}}^{+}\langle k+1\rangle + \boldsymbol{K}\nu \tag{H.6}$$

$$\hat{\boldsymbol{P}}\langle k+1\rangle = \hat{\boldsymbol{P}}^{+}\langle k+1\rangle - \boldsymbol{K}\boldsymbol{H}\hat{\boldsymbol{P}}^{+}\langle k+1\rangle \tag{H.7}$$

Uncertainty is now *decreased* or *deflated*, since new information, from the sensors, is being incorporated. The matrix

$$\boldsymbol{K} = \boldsymbol{P}^{+}\langle k+1\rangle\,\boldsymbol{H}^{T}\left(\underbrace{\boldsymbol{H}\,\boldsymbol{P}^{+}\langle k+1\rangle\,\boldsymbol{H}^{T} + \hat{\boldsymbol{W}}}\right)^{-1} \in \mathbb{R}^{n\times p} \tag{H.8}$$

is known as the Kalman gain. The term indicated is the estimated covariance of the innovation and comprises the uncertainty in the state and the estimated measurement noise covariance. If the innovation has high uncertainty in some dimensions then the Kalman gain will be correspondingly small, that is, if the new information is uncertain then only small changes are made to the state vector. The term $\boldsymbol{H}\boldsymbol{P}^{+}\langle k+1\rangle\,\boldsymbol{H}^{T}$ in Eq. H.13 *projects* the covariance of the state estimate into the space of sensor values.

The covariance matrix must be positive-definite but after many updates the accumulated numerical errors may cause this matrix to be no longer symmetric. The positive-definite structure can be enforced by using the Joseph form of Eq. H.7

$$\hat{\boldsymbol{P}}\langle k+1\rangle = \left(\boldsymbol{I}_{n\times n} - \boldsymbol{K}\boldsymbol{H}\right)\hat{\boldsymbol{P}}^{+}\langle k+1\rangle\left(\boldsymbol{I}_{n\times n} - \boldsymbol{K}\boldsymbol{H}\right)^{T} + \boldsymbol{K}\hat{\boldsymbol{V}}\boldsymbol{K}^{T}$$

but this is computationally more costly.

The equations above constitute the classical Kalman filter which is widely used in robotics, aerospace and econometric applications. The filter has a number of important characteristics. Firstly it is optimal, but only if the noise is truly Gaussian with zero mean and time invariant parameters. This is often a good assumption but not always. Secondly it is recursive, the output of one iteration is the input to the next. Thirdly, it is asynchronous. At a particular iteration if no sensor information is available we just perform the prediction step and not the update. In the case that there are different sensors, each with their own $\boldsymbol{H}$, and different sample rates, we just apply the update with the appropriate $\boldsymbol{z}$ and $\boldsymbol{H}$. The filter must be initialized with some reasonable value of $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{P}}$, as well as good choices of the covariance matrices $\hat{\boldsymbol{V}}$ and $\hat{\boldsymbol{W}}$. As the filter runs the estimated covariance $\|\hat{\boldsymbol{P}}\|$ decreases but never reaches zero – the minimum value can be shown to be a function of $\hat{\boldsymbol{V}}$ and $\hat{\boldsymbol{W}}$. The Kalman-Bucy filter is a continuous-time version of this filter.

The covariance matrix $\hat{\boldsymbol{P}}$ is rich in information. The diagonal elements $\hat{\boldsymbol{P}}_{ii}$ are the variance, or uncertainty, in the state $\boldsymbol{x}_i$. The off-diagonal elements $\hat{\boldsymbol{P}}_{ij}$ are the correlations between states $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ and indicate that the errors are not independent. The correlations are critical in allowing any piece of new information to *flow through* to adjust all the states that affect a particular process output.

H.2 Nonlinear Systems – Extended Kalman Filter

For the case where the system is not linear it can be described generally by two functions: the state transition (the motion model in robotics) and the sensor model

$$\boldsymbol{x}\langle k{+}1\rangle = f\big(\boldsymbol{x}\langle k\rangle, \boldsymbol{u}\langle k\rangle, \boldsymbol{v}\langle k\rangle\big) \tag{H.9}$$

$$\boldsymbol{z}\langle k\rangle = \boldsymbol{h}\big(\boldsymbol{x}\langle k\rangle, \boldsymbol{w}\langle k\rangle\big) \tag{H.10}$$

and as before we represent model uncertainty, external disturbances and sensor noise by Gaussian random variables $\boldsymbol{v}$ and $\boldsymbol{w}$.

We linearize the state transition function about the current state estimate $\hat{\boldsymbol{x}}_k$ as shown in Fig. H.2 resulting in

$$\boldsymbol{x}'\langle k{+}1\rangle \approx \boldsymbol{F}_x\boldsymbol{x}'\langle k\rangle + \boldsymbol{F}_u\boldsymbol{u}\langle k\rangle + \boldsymbol{F}_v\boldsymbol{v}\langle k\rangle \tag{H.11}$$

$$\boldsymbol{z}'\langle k\rangle \approx \boldsymbol{H}_x\boldsymbol{x}'\langle k\rangle + \boldsymbol{H}_w\boldsymbol{w}\langle k\rangle \tag{H.12}$$

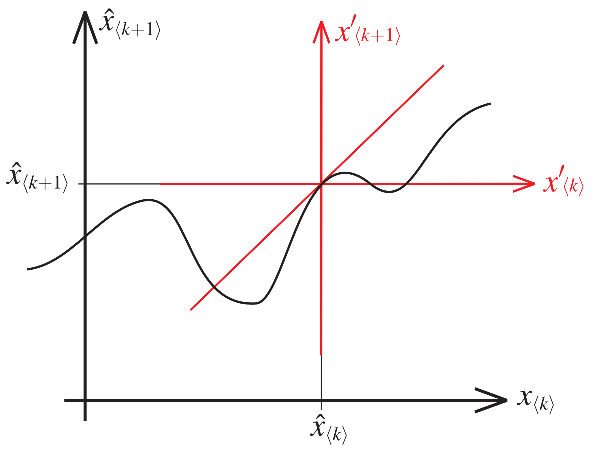
where $\boldsymbol{F}_x = \partial \boldsymbol{f}/\partial \boldsymbol{x} \in \mathbb{R}^{n\times n}$, $\boldsymbol{F}_u = \partial \boldsymbol{f}/\partial \boldsymbol{u} \in \mathbb{R}^{n\times m}$, $\boldsymbol{F}_v = \partial \boldsymbol{f}/\partial \boldsymbol{v} \in \mathbb{R}^{n\times n}$, $\boldsymbol{H}_x = \partial \boldsymbol{h}/\partial \boldsymbol{x} \in \mathbb{R}^{p\times n}$ and $\boldsymbol{H}_w = \partial \boldsymbol{h}/\partial \boldsymbol{w} \in \mathbb{R}^{p\times p}$ are Jacobians of the functions $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$. Equating coefficients between Eq. H.1 and Eq. H.11 gives $\boldsymbol{F} \sim \boldsymbol{F}_x$, $\boldsymbol{G} \sim \boldsymbol{F}_u$ and $\boldsymbol{v}\langle k\rangle \sim \boldsymbol{F}_v\boldsymbol{v}\langle k\rangle$; and between Eq. H.2 and Eq. H.12 gives $\boldsymbol{H} \sim \boldsymbol{H}_x$ and $\boldsymbol{w}\langle k\rangle \sim \boldsymbol{H}_w\boldsymbol{w}\langle k\rangle$.

Taking the prediction equation Eq. H.9 with $\boldsymbol{v}\langle k\rangle = 0$, and the covariance equation Eq. H.4 with the linearized terms substituted we can write the prediction step as

$$\hat{\boldsymbol{x}}^+\langle k{+}1\rangle = \boldsymbol{f}\big(\hat{\boldsymbol{x}}\langle k\rangle, \boldsymbol{u}\langle k\rangle\big)$$
$$\hat{\boldsymbol{P}}^+\langle k{+}1\rangle = \boldsymbol{F}_x\hat{\boldsymbol{P}}\langle k\rangle\boldsymbol{F}_x^T + \boldsymbol{F}_v\hat{\boldsymbol{V}}\boldsymbol{F}_v^T$$

and the update step as

$$\hat{\boldsymbol{x}}\langle k{+}1\rangle = \hat{\boldsymbol{x}}^+\langle k{+}1\rangle + \boldsymbol{K}\nu$$
$$\hat{\boldsymbol{P}}\langle k{+}1\rangle = \hat{\boldsymbol{P}}^+\langle k{+}1\rangle - \boldsymbol{K}\boldsymbol{H}_x\hat{\boldsymbol{P}}^+\langle k{+}1\rangle$$



$\hat{x}\langle k{+}1\rangle$
$x'\langle k{+}1\rangle$
$\hat{x}\langle k{+}1\rangle$
$x'\langle k\rangle$
$x\langle k\rangle$
$\hat{x}\langle k\rangle$

Fig. H.2. One dimensional example illustrating how the nonlinear state transition function $f: x_k \mapsto x_{k+1}$ shown in *black* is linearized about the point $(\hat{x}\langle k\rangle, \hat{x}\langle k{+}1\rangle)$ shown in *red*

```
Procedure EKF
Input : $\hat{x}\langle k\rangle \in \mathbb{R}^n$, $\hat{P}\langle k\rangle \in \mathbb{R}^{n\times n}$, $u\langle k\rangle \in \mathbb{R}^m$, $z\langle k+1\rangle \in \mathbb{R}^p$, $\hat{V} \in \mathbb{R}^{n\times n}$, $\hat{W} \in \mathbb{R}^{p\times p}$
Output: $\hat{x}\langle k+1\rangle \in \mathbb{R}^n$, $\hat{P}\langle k+1\rangle \in \mathbb{R}^{n\times n}$
– linearize about $x = \hat{x}\langle k\rangle$
compute Jacobians: $F_x \in \mathbb{R}^{n\times n}$, $F_v \in \mathbb{R}^{n\times n}$, $H_x \in \mathbb{R}^{p\times n}$, $H_w \in \mathbb{R}^{p\times p}$
– the prediction step
    $\hat{x}^+\langle k+1\rangle = f(\hat{x}\langle k\rangle, u\langle k\rangle)$ // predict state at next time step
    $\hat{P}^+\langle k+1\rangle = F_x \hat{P}\langle k\rangle F_x^T + F_v \hat{V} F_v^T$ // predict covariance at next time step
– the update step
    $\nu = z\langle k+1\rangle - h\left(\hat{x}^+\langle k+1\rangle\right)$ // innovation : measured – predicted sensor value
    $K = P^+\langle k+1\rangle H_x^T \left[ H_x P^+\langle k+1\rangle H_x^T + H_w \hat{W} H_w^T \right]^{-1}$ // Kalman gain
    $\hat{x}\langle k+1\rangle = \hat{x}^+\langle k+1\rangle + K\nu$ // update state estimate
    $\hat{P}\langle k+1\rangle = \hat{P}^+\langle k+1\rangle - K H_x \hat{P}^+\langle k+1\rangle$ // update covariance estimate
```

Algorithm H.1. Procedure EKF

where the Kalman gain is now

$$K = P^+\langle k+1\rangle H_x^T \left(H_x P^+\langle k+1\rangle H_x^T + H_w \hat{W} H_w^T \right)^{-1} \quad \text{(H.13)}$$

These equations are only valid at the linearization point $\hat{x}\langle k\rangle$ – the Jacobians F_x, F_v, H_x, H_w must be computed at every iteration.◀ The full procedure is summarized in Algorithm H.1.

Properly these matrices should be denoted as depending on the time step, i.e. $F_x\langle k\rangle$ but this has been dropped in the interest of readability.

A fundamental problem with the extended Kalman filter is that PDFs of the random variables are no longer Gaussian after being operated on by the nonlinear functions $f(\cdot)$ and $h(\cdot)$. We can easily illustrate this by considering a nonlinear scalar function $y = (x+2)^2/4$. We will draw a million Gaussian random numbers from the normal distribution $\mathcal{N}(5, 4)$ which has a mean of 5 and a standard deviation of 2

```
>> x = 2*randn(1000000,1) + 5;
```

and map them through our function

```
>> y = (x+2).^2 / 4;
```

and plot the probability density function of y

```
>> histogram(y, 'Normalization', 'pdf');
```

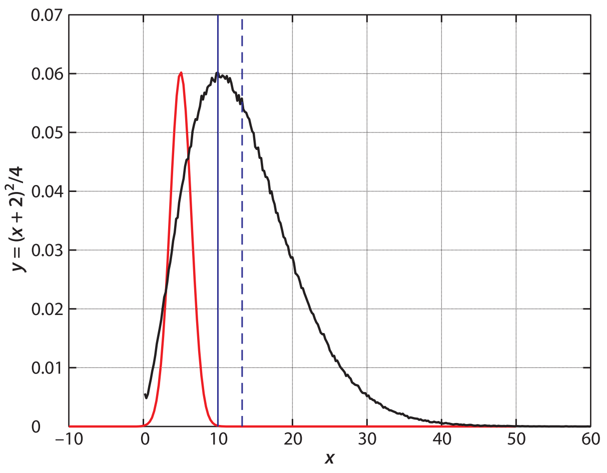


Fig. H.3. PDF of the state x (*red*) which is Gaussian $\mathcal{N}(5, 4)$ and the PDF of the nonlinear function $y = (x+2)^2/4$ (*black*). The peak and the mean of the nonlinear distribution are shown by *blue solid* and *dashed vertical lines* respectively

which is shown in Fig. H.3. We see that the PDF of y is substantially changed and no longer Gaussian. It has lost its symmetry so the mean value is greater than the mode. The Jacobians that appear in the EKF equations appropriately scale the covariance but the resulting non-Gaussian distribution breaks the assumptions which guarantee that the Kalman filter is an optimal estimator. Alternatives include the iterated EKF described by Jazwinski (2007) or the Unscented Kalman Filter (UKF) (Julier and Uhlmann 2004) or the sigma-point filter which uses discrete sample points (sigma points) to approximate the PDF.

I Graphs

A graph is an abstract representation of a set of objects connected by links and depicted graphically as shown in Fig. I.1. Mathematically a graph is denoted $G(V, E)$ where V are the vertices or nodes, and E are the links that connect pairs of vertices and are called edges or arcs. Edges can be directed (*arrows*) or undirected as in this case. Edges can have an associated weight or cost associated with moving from one vertex to another. A sequence of edges from one vertex to another is a path, and a sequence that starts and ends at the same vertex is a cycle. An edge from a vertex to itself is a loop. Graphs can be used to represent transport, communications or social networks, and this branch of mathematics is graph theory.

The Toolbox provides a MATLAB® graph class called PGraph that supports embedded graphs where the vertices are associated with a point in an n-dimensional space. To create a new graph

This class is used other Toolbox classes such as PRM, Lattice, RRT, PoseGraph and BundleAdjust. MATLAB 2015b introduced a built in graph class to represent graphs.

```
>> g = PGraph()
g =
  2 dimensions
  0 vertices
  0 edges
  0 components
```

and by default the nodes of the graph exist in a 2-dimensional space. We can add nodes to the graph

```
>> g.add_node( rand(2,1) );
>> g.add_node( rand(2,1) );
>> g.add_node( rand(2,1) );
>> g.add_node( rand(2,1) );
>> g.add_node( rand(2,1) );
```

and each has a random coordinate. The add_node method returns an integer identifier for the node just added. A summary of the graph is given with its display method

```
>> g
g =
  2 dimensions
  5 vertices
  0 edges
  0 components
```

and shows that the graph has 5 nodes but no edges. The nodes are numbered 1 to 5 and we add edges between pairs of nodes

```
>> g.add_edge(1, 2);
>> g.add_edge(1, 3);
>> g.add_edge(1, 4);
>> g.add_edge(2, 3);
>> g.add_edge(2, 4);
>> g.add_edge(4, 5);
>> g
g =
  2 dimensions
  5 vertices
  6 edges
  1 components
```

By default the distance between the nodes is the Euclidean distance between the vertices but this can be overridden by a third argument to `add_edge`. The methods `add_node` and `add_edge` return an integer that uniquely identifies the node or edge just created. The graph has one component, that is all the nodes are connected into one network. The graph can be plotted by

```
>> g.plot('labels')
```

as shown in Fig. I.1. The vertices are shown as blue circles, and the option `'labels'` displays the vertex index next to the circle. Edges are shown as black lines joining vertices. Many options exist to change default plotting behavior. Note that only graphs embedded in 2- and 3-dimensional space can be plotted.

The neighbors of vertex 2 are

```
>> g.neighbours(2)
ans =
     3     4     1
```

which are vertices connected to vertex 2 by edges. Each edge has a unique index and the edges connecting to vertex 2 are

```
>> e = g.edges(2)
e =
     4     5     1
```

The cost or length of these edges is

```
>> g.cost(e)
ans =
    0.9597    0.3966    0.6878
```

and clearly edge 5 has a lower cost than edges 4 and 1. Edge 5

```
>> g.vertices(5)'
ans =
     2     4
```

joins vertices 2 and 4, and vertex 4 is clearly the closest neighbor of vertex 2. Frequently we wish to obtain a node's neighboring vertices and their distances at the same time, and this can be achieved conveniently by

```
>> [n,c] = g.neighbours(2)
n =
     3     4     1
c =
    0.9597    0.3966    0.6878
```

Concise information about a node can be obtained by

```
>> g.about(1)
Node 1 #1@ (0.814724 0.905792 )
  neighbours:   >-o-> 2 3 4
  edges:   >-o-> 1 2 3
```

Arbitrary data can be attached to any node or edge by the methods `setvdata` and `setedata` respectively and retrieved by the methods `vdata` and `edata` respectively.

The vertex closest to the coordinate (0.5, 0.5) is

```
>> g.closest([0.5, 0.5])
ans =
     4
```

and the vertex closest to an interactively selected point is given by `g.pick`.

The minimum cost path between any two nodes in the graph can be computed using well known algorithms such as A* (Nilsson 1971)

```
>> g.Astar(3, 5)
ans =
     3     2     4     5
```

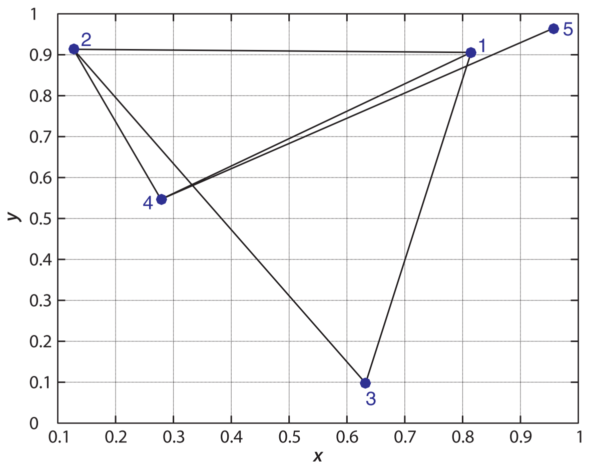


Fig. I.1.
An example graph generated by the `PGraph` class

or the earlier method by Dijstrka (1959). By default the graph is treated as undirected, that is, the edges have no preferred direction. The `'directed'` option causes edges to be treated as directed, and the path will only traverse edges in their specified direction which is from the first to the second argument of the method `add_edge`.

Methods exist to compute various other representations of the graph such as adjacency, incidence, degree and Laplacian matrices.

J Peak Finding

A commonly encountered problem is estimating the position of the peak of some discrete 1-dimensional signal $y(k)$, $k \in \mathbb{Z}$, see for example Fig. J.1a

```
>> load peakfit1
>> plot(y,  '-o')
```

Finding the peak to the nearest integer is straightforward using MATLAB's `max` function

```
>> [ypk,k] = max(y)
ypk =
    0.9905
k =
     8
```

which indicates the peak occurs at the eighth element and has a value of 0.9905. In this case there is more than one peak and we can use the Toolbox function `peak` instead

```
>> [ypk,k] = peak(y)
ypk =
    0.9905    0.6718   -0.5799
k =
     8    25    16
```

which has returned three maxima in descending magnitude. A common test of the quality of a peak is its magnitude and the ratio of the height of the second peak to the first peak

```
>> ypk(2)/ypk(1)
ans =
    0.6783
```

which is called the ambiguity ratio and is ideally small.

This signal is a sampled representation of a continuous underlying signal $y(x)$ and the real peak might actually lie between the samples. If we look at a zoomed version of the signal, Fig. J.1b, we can see that although the eighth point is the maximum the

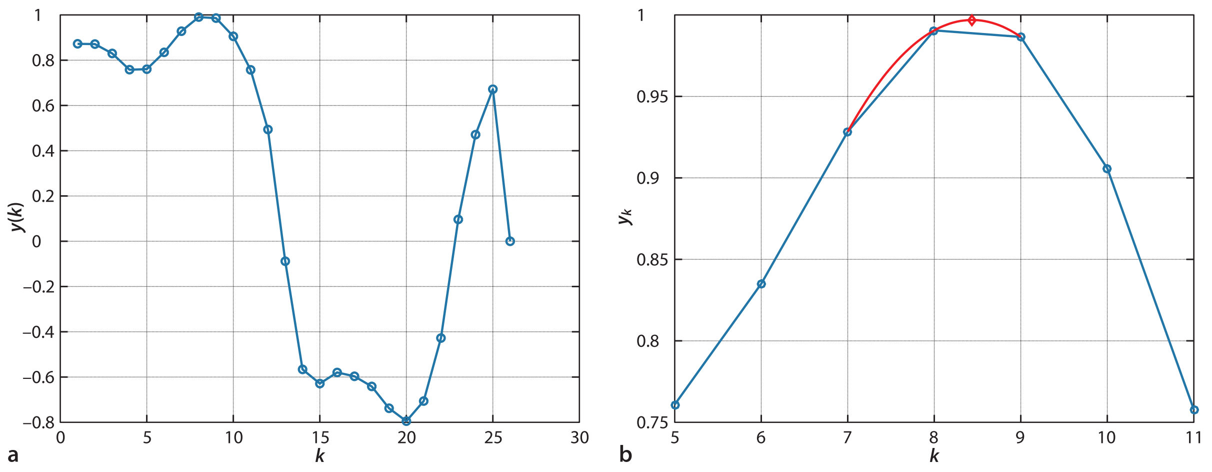


Fig. J.1. Peak fitting. **a** A signal with several local maxima; **b** closeup view of the first maxima with the fit curve (*red*) and the estimated peak (*red*-◊)

ninth point is only slightly lower so the peak lies somewhere between points eight and nine. A common approach is to fit a parabola

$$y = a\delta_x^2 + b\delta_x + c, \quad \delta_x \in \mathbb{R} \tag{J.1}$$

to the points surrounding the peak. For the discrete peak that occurs at (k, y_k) then $\delta_x = 0$ corresponds to k and the discrete x-coordinates on either side correspond to $\delta_x = -1$ and $\delta_x = +1$ respectively. Substituting the points $(k-1, y_{k-1})$, (k, y_k) and $(k+1, y_{k+1})$ into Eq. J.1 we can write three equations

$$\begin{aligned} y_{k-1} &= a - b + c \\ y_k &= c \\ y_{k+1} &= a + b + c \end{aligned}$$

or in compact matrix form as

$$\begin{pmatrix} y_{k-1} \\ y_k \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

and then solve for the parabolic coefficients

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} y_{k-1} \\ y_k \\ y_{k+1} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & -2 & 1 \\ -1 & 0 & 1 \\ 0 & 2 & 0 \end{pmatrix} \begin{pmatrix} y_{k-1} \\ y_k \\ y_{k+1} \end{pmatrix} \tag{J.2}$$

The maxima of the parabola occurs when its derivative is zero

$$2a\delta_x + b = 0$$

and substituting the values of a and b from Eq. J.2 we find the displacement of the peak of the fitted parabola with respect to the discrete maxima

$$\delta_x = \frac{1}{2} \frac{y_{k-1} - y_{k+1}}{y_{k-1} - 2y_k + y_{k+1}}, \; \delta_x \in (-1, 1)$$

so the refined, or interpolated, position of the maxima is at

$$\hat{x} = k + \delta_x \in \mathbb{R}$$

and the estimated value of the maxima is obtained by substituting δ_x into Eq. J.1.

The coefficient a, which is negative for a maxima, indicates the sharpness of the peak which can be useful in determining whether a peak is *sufficiently* sharp. A large magnitude of a indicates a well defined sharp peak wheras a low value indicates a very broad peak for which estimation of a refined peak may not be so accurate.

Continuing the earlier example we can use the Toolbox function `peak` to estimate the refined peak positions

```
>> [ymax,xmax] = peak(y, 'interp', 2)
ymax =
    0.9905    0.6718   -0.5799
xmax =
    8.4394   24.7299   16.2438
```

where the argument after the `'interp'` option indicates that a second-order polynomial should be fitted. The fitted parabola is shown in red in Fig. J.1b and is plotted if the option `'plot'` is given.

If the signal has superimposed noise then there are likely to be multiple peaks, many of which are quite minor, and this can be overcome by specifying the *scale* of the peak. For example the peaks that are greater than all other values within ±5 values in the horizontal direction are

```
>> peak(y, 'scale', 5)
ans =
    0.9905    0.8730    0.6718
```

In this case the result is unchanged since the signal is fairly smooth.

For a 2-dimensional signal we follow a similar procedure but instead fit a *paraboloid*

$$z = ax^2 + by^2 + cx + dy + e \tag{J.3}$$

which has five coefficients that can be calculated from the center value (the discrete maximum) and its four neighbors (north, south, east and west) using a similar procedure to above. The displacement of the estimated peak with respect to the central point is

$$\delta_x = \frac{1}{2}\frac{z_e - z_w}{2z_c - z_w - z_e}, \quad \delta_x \in (-1, 1)$$
$$\delta_y = \frac{1}{2}\frac{z_s - z_n}{2z_c - z_n - z_s} \quad \delta_y \in (-1, 1)$$

In this case the coefficients a and b represent the sharpness of the peak in the x- and y-directions, and the quality of the peak can be considered as being $\min(a, b)$.

A 2D discrete signal was loaded from `peakfit1` earlier

```
>> z
z =
   -0.0696    0.0348    0.1394    0.2436    0.3480
    0.0800    0.2000    0.3202    0.4400    0.5600
    0.0400    0.1717    0.3662    0.4117    0.5200
    0.0002    0.2062    0.8766    0.4462    0.4802
   -0.0400    0.0917    0.2862    0.3317    0.4400
   -0.0800    0.0400    0.1602    0.2800    0.4000
```

In this small example it is clear that the peak is at element (3, 4) using image coordinate convention, but programatically this is

```
>> [zmax,i] = max(z(:))
zmax =
    0.8766
i =
    16
```

and the maximum is at the sixteenth element in row-major order◂ which we convert to array subscripts

Counting the elements, starting with 1 at the top-left down each column then back to the top of the next rightmost column.

```
>> [y,x] = ind2sub(size(z), i)
y =
     4
x =
     3
```

We can find this more conveniently using the Toolbox function `peak2`

```
>> [zpk,xy]=peak2(z)
zpk =
    0.8766    0.5600
xy =
     3     5
     4     2
```

which has returned two local maxima, one per column of the returned variables. This function will return all nonlocal maxima where the size of the local region is given by the `'scale'` option. As for the 1-dimensional case we can refine the estimate of the peak

```
>> [zpk,xy]=peak2(z, 'interp')
Warning: Peak at (5,2) too close to edge of image
zpk =
    0.8839
xy =
    3.1090
    3.9637
```

that is, the peak is at element (3.1090, 3.9637). When this process is applied to image data it is referred to as subpixel interpolation.

Bibliography

Achtelik MW (2014) Advanced closed loop visual navigation for micro aerial vehicles. Ph.D. thesis, ETH Zurich

Agarwal S, Furukawa Y, Snavely N, Simon I, Curless B, Seitz SM, Szeliski R (2011) Building Rome in a day. Commun ACM 54(10):105–112

Agarwal P, Burgard W, Stachniss C (2014) Survey of geodetic mapping methods: Geodetic approaches to mapping and the relationship to graph-based SLAM. IEEE Robot Autom Mag 21(3):63–80

Agrawal M, Konolige K, Blas M (2008) CenSurE: Center surround extremas for realtime feature detection and matching. In: Forsyth D, Torr P, Zisserman A (eds) Lecture notes in computer science. Computer Vision – ECCV 2008, vol 5 305. Springer-Verlag, Berlin Heidelberg, pp 102–115

Albertos P, Mareels I (2010) Feedback and control for everyone. Springer-Verlag, Berlin Heidelberg

Altmann SL (1989) Hamilton, Rodrigues, and the quaternion scandal. Math Mag 62(5):291–308

Alton K, Mitchell IM (2006) Optimal path planning under defferent norms in continuous state spaces. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp 866–872

Andersen N, Ravn O, Sørensen A (1993) Real-time vision based control of servomechanical systems. In: Chatila R, Hirzinger G (eds) Lecture notes in control and information sciences. Experimental Robotics II, vol 190. Springer-Verlag, Berlin Heidelberg, pp 388–402

Andersson RL (1989) Dynamic sensing in a ping-pong playing robot. IEEE T Robotic Autom 5(6):728–739

Antonelli G (2014) Underwater robots: Motion and force control of vehicle-manipulator systems, 3rd ed. Springer Tracts in Advanced Robotics, vol 2. Springer-Verlag, Berlin Heidelberg

Arandjelovi R, Zisserman A (2012) Three things everyone should know to improve object retrieval. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp 2911–2918

Arkin RC (1999) Behavior-based robotics. MIT Press, Cambridge, Massachusetts

Armstrong WW (1979) Recursive solution to the equations of motion of an N-link manipulator. In: Proceedings of the 5th World Congress on Theory of Machines and Mechanisms, Montreal, Jul, pp 1 343–1 346

Armstrong BS (1988) Dynamics for robot control: Friction modelling and ensuring excitation during parameter identification. Stanford University

Armstrong B (1989) On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics. Int J Robot Res 8(6):28

Armstrong B, Khatib O, Burdick J (1986) The explicit dynamic model and inertial parameters of the Puma 560 Arm. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), vol 3. pp 510–518

Armstrong-Hélouvry B, Dupont P, De Wit CC (1994) A survey of models, analysis tools and compensation methods for the control of machines with friction. Automatica 30(7):1 083–1 138

Arun KS, Huang TS, Blostein SD (1987) Least-squares fitting of 2 3-D point sets. IEEE T Pattern Anal 9(5):699–700

Asada H (1983) A geometrical representation of manipulator dynamics and its application to arm design. J Dyn Syst-T ASME 105:131

Astolfi A (1999) Exponential stabilization of a wheeled mobile robot via discontinuous control. J Dyn Syst-T ASME 121(1):121–126

Azarbayejani A, Pentland AP (1995) Recursive estimation of motion, structure, and focal length. IEEE T Pattern Anal 17(6):562–575

Bailey T (n.d.) Software resources. University of Sydney. http://www-personal.acfr.usyd.edu.au/tbailey

Bailey T, Durrant-Whyte H (2006) Simultaneous localization and mapping: Part II. IEEE Robot Autom Mag 13(3):108–117

Bakthavatchalam M, Chaumette F, Tahri O (2015) An improved modelling scheme for photometric moments with inclusion of spatial weights for visual servoing with partial appearance/disappearance. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp 6037–6043

Baldridge AM, Hook SJ, Grove CI, Rivera G (2009) The ASTER spectral library version 2.0. Remote Sens Environ 113(4):711–715

Ball RS (1876) The theory of screws: A study in the dynamics of a rigid body. Hodges, Foster & Co., Dublin

Ball RS (1908) A treatise on spherical astronomy. Cambridge University Press, New York

Ballard DH (1981) Generalizing the Hough transform to detect arbitrary shapes. Pattern Recogn 13(2):111–122

Banks J, Corke PI (2001) Quantitative evaluation of matching methods and validity measures for stereo vision. Int J Robot Res 20(7):512–532

Bar-Shalom Y, Fortmann T (1988) Tracking and data association. Mathematics in science and engineering, vol 182. Academic Press, London Oxford

Bar-Shalom Y, Rong Li X, Thiagalingam Kirubarajan (2001) Estimation with applications to tracking and navigation. John Wiley & Sons, Inc., Chichester

Bauer J, Sünderhauf N, Protzel P (2007) Comparing several implementations of two recently published feature detectors. In: IFAC Symposium on Intelligent Autonomous Vehicles (IAV). Toulouse

Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-up robust features (SURF). Comput Vis Image Und 110(3):346–359

Benosman R, Kang SB (2001) Panoramic vision: Sensors, theory, and applications. Springer-Verlag, Berlin Heidelberg

Benson KB (ed) (1986) Television engineering handbook. McGraw-Hill, New York

Bertozzi M, Broggi A, Cardarelli E, Fedriga R, Mazzei L, Porta P (2011) VIAC expedition: Toward autonomous mobility. IEEE Robot Autom Mag 18(3):120–124

Besl PJ, McKay HD (1992) A method for registration of 3-D shapes. IEEE T Pattern Anal 14(2): 239–256

Bhat DN, Nayar SK (2002) Ordinal measures for image correspondence. IEEE T Pattern Anal 20(4): 415–423

Biber P, Straßer W (2003) The normal distributions transform: A new approach to laser scan matching. In: Proceedings of the IEEE/RSJ International Conference on intelligent robots and systems (IROS), vol 3. pp 2743–2748

Bishop CM (2006) Pattern recognition and machine learning. Information science and statistics. Springer-Verlag, New York

Blewitt M (2011) Celestial navigation for yachtsmen. Adlard Coles Nautical, London

Bolles RC, Baker HH, Marimont DH (1987) Epipolar-plane image analysis: An approach to determining structure from motion. Int J Comput Vision 1(1):7–55, Mar

Bolles RC, Baker HH, Hannah MJ (1993) The JISCT stereo evaluation. In: Image Understanding Workshop: proceedings of a workshop held in Washington, DC apr 18–21, 1993. Morgan Kaufmann, San Francisco, pp 263

Bolton W (2015) Mechatronics: Electronic control systems in mechanical and electrical engineering, 6th ed. Pearson, Harlow

Borenstein J, Everett HR, Feng L (1996) Navigating mobile robots: Systems and techniques. AK Peters, Ltd. Natick, MA, USA, Out of print and available at http://www-personal.umich.edu/˜johannb/Papers/pos96rep.pdf

Borgefors G (1986) Distance transformations in digital images. Comput Vision Graph 34(3):344–371

Bostrom N (2016) Superintelligence: Paths, dangers, strategies. Oxford University Press, Oxford, 432 p

Bouguet J-Y (2010) Camera calibration toolbox for MATLAB®. http://www.vision.caltech.edu/bouguetj/calib_doc

Brady M, Hollerbach JM, Johnson TL, Lozano-Pérez T, Mason MT (eds) (1982) Robot motion: Planning and control. MIT Press, Cambridge, Massachusetts

Braitenberg V (1986) Vehicles: Experiments in synthetic psychology. MIT Press, Cambridge, Massachusetts

Bray H (2014) You are here: From the compass to GPS, the history and future of how we find ourselves. Basic Books, New York

Brockett RW (1983) Asymptotic stability and feedback stabilization. In: Brockett RW, Millmann RS, Sussmann HJ (eds) Progress in mathematics. Differential geometric control theory, vol 27. pp 181–191

Broida TJ, Chandrashekhar S, Chellappa R (1990) Recursive 3-D motion estimation from a monocular image sequence. IEEE T Aero Elec Sys 26(4):639–656

Brooks RA (1986) A robust layered control system for a mobile robot. IEEE T Robotic Autom 2(1):14–23

Brooks RA (1989) A robot that walks: Emergent behaviors from a carefully evolved network. MIT AI Lab, Memo 1091

Brown MZ, Burschka D, Hager GD (2003) Advances in computational stereo. IEEE T Pattern Anal 25(8):993–1 008

Brynjolfsson E, McAfee A (2014) The second machine age: Work, progress, and prosperity in a time of brilliant technologies. W. W. Norton & Co., New York

Buehler M, Iagnemma K, Singh S (eds) (2007) The 2005 DARPA grand challenge: The great robot race. Springer Tracts in Advanced Robotics, vol 36. Springer-Verlag, Berlin Heidelberg

Buehler M, Iagnemma K, Singh S (eds) (2010) The DARPA urban challenge. Tracts in Advanced Robotics, vol 56. Springer-Verlag, Berlin Heidelberg

Bukowski R, Haynes LS, Geng Z, Coleman N, Santucci A, Lam K, Paz A, May R, DeVito M (1991) Robot hand-eye coordination rapid prototyping environment. In: Proc ISIR, pp 16.15–16.28

Buttazzo GC, Allotta B, Fanizza FP (1993) Mousebuster: A robot system for catching fast moving objects by vision. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Atlanta, pp 932–937

Calonder M, Lepetit V, Strecha C, Fua P (2010) BRIEF: Binary robust independent elementary features. In: Daniilidis K, Maragos P, Paragios N (eds) Lecture notes in computer science. Computer Vision – ECCV 2010, vol 6 311. Springer-Verlag, Berlin Heidelberg, pp 778–792

Canny JF (1983) Finding edges and lines in images. MIT, Artificial Intelligence Laboratory, AI-TR-720. Cambridge, MA

Canny J (1987) A computational approach to edge detection. In: Fischler MA, Firschein O (eds) Readings in computer vision: Issues, problems, principles, and paradigms. Morgan Kaufmann, San Francisco, pp 184–203

Censi A (2008) An ICP variant using a point-to-line metric. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp 19–25

Chahl JS, Srinivasan MV (1997) Reflective surfaces for panoramic imaging. Appl Optics 31(36):8 275–8 285

Chatfield K, Lempitsky VS, Vedaldi A, Zisserman A (2011) The devil is in the details: An evaluation of recent feature encoding methods. In: Proceedings of the British Machine Vision Conference 2011. 12 p

Chaumette F (1990) La relation vision-commande: Théorie et application et des tâches robotiques. Ph.D. thesis, Université de Rennes 1

Chaumette F (1998) Potential problems of stability and convergence in image-based and position-based visual servoing. In: Kriegman DJ, Hager GD, Morse AS (eds) Lecture notes in control and information sciences. The confluence of vision and control, vol 237. Springer-Verlag, Berlin Heidelberg, pp 66–78

Chaumette F (2004) Image moments: A general and useful set of features for visual servoing. IEEE T Robotic Autom 20(4):713–723

Chaumette F, Hutchinson S (2006) Visual servo control 1: Basic approaches. IEEE Robot Autom Mag 13(4):82–90

Chaumette F, Hutchinson S (2007) Visual servo control 2: Advanced approaches. IEEE Robot Autom Mag 14(1):109–118

Chaumette F, Rives P, Espiau B (1991) Positioning of a robot with respect to an object, tracking it and estimating its velocity by visual servoing. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Seoul, pp 2 248–2 253

Chesi G, Hashimoto K (eds) (2010) Visual servoing via advanced numerical methods. Lecture notes in computer science, vol 401. Springer-Verlag, Berlin Heidelberg

Chiaverini S, Sciavicco L, Siciliano B (1991) Control of robotic systems through singularities. Lecture notes in control and information sciences. Advanced Robot Control, Proceedings of the International Workshop on Nonlinear and Adaptive Control: Issues in Robotics, vol 162. Springer-Verlag, Berlin Heidelberg, pp 285–295

Chiuso A, Favaro P, Jin H, Soatto S (2002) Structure from motion causally integrated over time. IEEE T Pattern Anal 24(4):523–535

Choset HM, Lynch KM, Hutchinson S, Kantor G, Burgard W, Kavraki LE, Thrun S (2005) Principles of robot motion. MIT Press, Cambridge, Massachusetts

Colicchia G, Waltner C, Hopf M, Wiesner H (2009) The scallop's eye – A concave mirror in the context of biology. Physics Education 44(2):175–179

Collewet C, Marchand E, Chaumette F (2008) Visual servoing set free from image processing. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA). pp 81–86

Commission Internationale de L'Éclairage (1987) Colorimetry, 2nd ed. Commission Internationale de L'Eclairage, CIE No 15.2

Corke PI (1994) High-performance visual closed-loop robot control. University of Melbourne, Dept. Mechanical and Manufacturing Engineering. http://eprints.unimelb.edu.au/archive/00000547/01/thesis.pdf

Corke PI (1996a) In situ measurement of robot motor electrical constants. Robotica 14(4):433–436

Corke PI (1996b) Visual control of robots: High-performance visual servoing. Mechatronics, vol 2. Research Studies Press (John Wiley). Out of print and available at http://www.petercorke.com/bluebook

Corke PI (2001) Mobile robot navigation as a planar visual servoing problem. In: Jarvis RA, Zelinsky A (eds) Springer tracts in advanced robotics. Robotics Research: The 10th International Symposium, vol 6. IFRR, Lorne, pp 361–372

Corke PI (2007) A simple and systematic approach to assigning Denavit-Hartenberg parameters. IEEE T Robotic Autom 23(3):590–594

Corke PI (2010) Spherical image-based visual servo and structure estimation. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Anchorage, pp 5 550–5 555

Corke PI, Armstrong-Hélouvry BS (1994) A search for consensus among model parameters reported for the PUMA 560 robot. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). San Diego, pp 1 608–1 613

Corke PI, Armstrong-Hélouvry BS (1995) A meta-study of PUMA 560 dynamics: A critical appraisal of literature data. Robotica 13(3):253–258

Corke PI, Good MC (1992) Dynamic effects in high-performance visual servoing. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Nice, pp 1 838–1 843

Corke PI, Good MC (1996) Dynamic effects in visual closed-loop systems. IEEE T Robotic Autom 12(5):671–683

Corke PI, Hutchinson SA (2001) A new partitioned approach to image-based visual servo control. IEEE T Robotic Autom 17(4):507–515

Corke PI, Dunn PA, Banks JE (1999) Frame-rate stereopsis using non-parametric transforms and programmable logic. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Detroit, pp 1928–1933

Corke PI, Strelow D, Singh S (2004) Omnidirectional visual odometry for a planetary rover. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS). Sendai, pp 4 007–4 012

Corke PI, Spindler F, Chaumette F (2009) Combining Cartesian and polar coordinates in IBVS. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS). St. Louis, pp 5 962–5 967

Corke PI, Paul R, Churchill W, Newman P (2013) Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp 2085–2092

Craig JJ (1987) Adaptive control of mechanical manipulators. Addison-Wesley Longman Publishing Co., Inc. Boston

Craig JJ (2005) Introduction to robotics: Mechanics and control, 3rd ed. Pearson/Prentice Hall, Upper Saddle River, New Jersey

Craig JJ, Hsu P, Sastry SS (1987) Adaptive control of mechanical manipulators. Int J Robot Res 6(2):16–28

Crombez N, Caron G, Mouaddib EM (2015) Photometric Gaussian mixtures based visual servoing. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp 5486–5491

Crone RA (1999) A history of color: The evolution of theories of light and color. Kluwer Academic, Dordrecht

Cummins M, Newman P (2008) FAB-MAP: Probabilistic localization and mapping in the space of appearance. Int J Robot Res 27(6):647

Cutting JE (1997) How the eye measures reality and virtual reality. Behav Res Meth Ins C 29(1):27–36

Daniilidis K, Klette R (eds) (2006) Imaging beyond the pinhole camera. Computational Imaging, vol 33. Springer-Verlag, Berlin Heidelberg

Dansereau DG (2014) Plenoptic signal processing for robust vision in field robotics. Ph.D. thesis, The University of Sydney

Davison AJ, Reid ID, Molton ND, Stasse O (2007) MonoSLAM: Real-time single camera SLAM. IEEE T Pattern Anal 29(6):1 052–1 067

Deguchi K (1998) Optimal motion control for image-based visual servoing by decoupling translation and rotation. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS). Victoria, Canada, pp 705–711

Dellaert F, Kaess M (2006) Square root SAM: Simultaneous localization and mapping via square root information smoothing. Int J Robot Res 25(12):1181–1203

Dellaert F, Seitz SM, Thorpe CE, Thrun S (2000) Structure from motion without correspondence. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Hilton Head Island, SC, pp 557–564

DeMenthon D, Davis LS (1992) Exact and approximate solutions of the perspective-three-point problem. IEEE T Pattern Anal 14(11):1 100–1 105

Denavit J, Hartenberg RS (1955) A kinematic notation for lower-pair mechanisms based on matrices. J Appl Mech-T ASME 22(1):215–221

Deo AS, Walker ID (1995) Overview of damped least-squares methods for inverse kinematics of robot manipulators. J Intell Robot Syst 14(1):43–68

Deriche R, Giraudon G (1993) A computational approach for corner and vertex detection. Int J Comput Vision 10(2):101–124

DeWitt BA, Wolf PR (2000) Elements of photogrammetry (with applications in GIS). McGraw-Hill, New York

Dickmanns ED (2007) Dynamic vision for perception and control of motion. Springer-Verlag, London

Dickmanns ED, Graefe V (1988a) Applications of dynamic monocular machine vision. Mach Vision Appl 1:241–261

Dickmanns ED, Graefe V (1988b) Dynamic monocular machine vision. Mach Vision Appl 1(4):223–240

Dickmanns ED, Zapp A (1987) Autonomous high speed road vehicle guidance by computer vision. In: Tenth Triennial World Congress of the International Federation of Automatic Control, vol 4. Munich, pp 221–226

Dijkstra EW (1959) A note on two problems in connexion with graphs. Numer Math 1(1):269–271

Dougherty ER, Lotufo RA (2003) Hands-on morphological image processing. Society of Photo-Optical Instrumentation Engineers (SPIE)

Duda RO, Hart PE (1972) Use of the Hough transformation to detect lines and curves in pictures. Commun ACM 15(1):11–15

Durrant-Whyte H, Bailey T (2006) Simultaneous localization and mapping: Part I. IEEE Robot Autom Mag 13(2):99–110

Espiau B, Chaumette F, Rives P (1992) A new approach to visual servoing in robotics. IEEE T Robotic Autom 8(3):313–326

Everett HR (1995) Sensors for mobile robots: Theory and application. AK Peters Ltd., Wellesley

Faugeras OD (1993) Three-dimensional computer vision: A geometric viewpoint. MIT Press, Cambridge, Massachusetts

Faugeras OD, Lustman F (1988) Motion and structure from motion in a piecewise planar environment. Int J Pattern Recogn 2(3):485–508

Faugeras O, Luong QT, Papadopoulou T (2001) The geometry of multiple images: The laws that govern the formation of images of a scene and some of their applications. MIT Press, Cambridge, Massachusetts

Featherstone R (1987) Robot dynamics algorithms. Kluwer Academic, Dordrecht

Feddema JT (1989) Real time visual feedback control for hand-eye coordinated robotic systems. Purdue University

Feddema JT, Mitchell OR (1989) Vision-guided servoing with feature-based trajectory generation. IEEE T Robotic Autom 5(5):691–700

Feddema JT, Lee CSG, Mitchell OR (1991) Weighted selection of image features for resolved rate visual feedback control. IEEE T Robotic Autom 7(1):31–47

Felzenszwalb PF, Huttenlocher DP (2004) Efficient graph-based image segmentation. Int J Comput Vision 59(2):167–181

Ferguson D, Stentz A (2006) Using interpolation to improve path planning: The Field D* algorithm. J Field Robotics 23(2):79–101

Fischler MA, Bolles RC (1981) Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 24(6):381–395

Flusser J (2000) On the independence of rotation moment invariants. Pattern Recogn 33(9):1405–1410

Fomena R, Chaumette F (2007) Visual servoing from spheres using a spherical projection model. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Rome, pp 2 080–2 085

Ford M (2015) Rise of the robots: Technology and the threat of a jobless future. Basic Books, New York

Förstner W (1994) A framework for low level feature extraction. In: Ecklundh J-O (ed) Lecture notes in computer science. Computer Vision – ECCV 1994, vol 800. Springer-Verlag, Berlin Heidelberg, pp 383–394

Förstner W, Gülch E (1987) A fast operator for detection and precise location of distinct points, corners and centres of circular features. In: ISPRS Intercommission Workshop. Interlaken, pp 149–155

Forsyth DA, Ponce J (2011) Computer vision: A modern approach, 2nd ed. Pearson, London

Fraundorfer F, Scaramuzza D (2012) Visual odometry: Part II – Matching, robustness, optimization, and applications. IEEE Robot Autom Mag 19(2):78–90

Freeman H (1974) Computer processing of line-drawing images. ACM Comput Surv 6(1):57–97

Friedman DP, Felleisen M, Bibby D (1987) The little LISPer. MIT Press, Cambridge, Massachusetts

Funda J, Taylor RH, Paul RP (1990) On homogeneous transforms, quaternions, and computational efficiency. IEEE T Robotic Autom 6(3):382–388

Gans NR, Hutchinson SA, Corke PI (2003) Performance tests for visual servo control systems, with application to partitioned approaches to visual servo control. Int J Robot Res 22(10–11):955

Gautier M, Khalil W (1992) Exciting trajectories for the identification of base inertial parameters of robots. Int J Robot Res 11(4):362

Geiger A, Roser M, Urtasun R (2010) Efficient large-scale stereo matching. In: Kimmel R, Klette R, Sugimoto A (eds) Computer vision – ACCV 2010: 10th Asian Conference on Computer Vision, Queenstown, New Zealand, November 8–12, 2010, revised selected papers, part I. Springer-Verlag, Berlin Heidelberg, pp 25–38

Geraerts R, Overmars MH (2004) A comparative study of probabilistic roadmap planners. In: Boissonnat J-D, Burdick J, Goldberg K, Hutchinson S (eds) Springer tracts in advanced robotics. Algorithmic Foundations of Robotics V, vol 7. Springer-Verlag, Berlin Heidelberg, pp 43–58

Gevers T, Gijsenij A, van de Weijer J, Geusebroek J-M (2012) Color in computer vision: Fundamentals and applications. John Wiley & Sons, Inc., Chichester

Geyer C, Daniilidis K (2000) A unifying theory for central panoramic systems and practical implications. In: Vernon D (ed) Lecture notes in computer science. Computer vision – ECCV 2000, vol 1 843. Springer-Verlag, Berlin Heidelberg, pp 445–461

Glover A, Maddern W, Warren M, Reid S, Milford M, Wyeth G (2012) OpenFABMAP: An open source toolbox for appearance-based loop closure detection. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp 4730–4735

Gonzalez R, Woods R (2008) Digital image processing, 3rd ed. Prentice Hall, Upper Saddle River, New Jersey

Gonzalez R, Woods R, Eddins S (2009) Digital image processing using MATLAB, 2nd ed. Gatesmark Publishing
Grassia FS (1998) Practical parameterization of rotations using the exponential map. Journal of Graphics Tools 3(3):29–48
Gregory RL (1997) Eye and brain: The psychology of seeing. Princeton University Press, Princeton, New Jersey
Grey CGP (2014) Humans need not apply. YouTube video, www.youtube.com/watch?v=7Pq-S557XQU
Grisetti G (n.d.) Teaching resources. Sapienza University of Rome. http://www.dis.uniroma1.it/~grisetti/teaching.html
Groves PD (2013) Principles of GNSS, inertial, and multisensor integrated navigation systems, 2nd ed. Artech House, Norwood, USA
Hager GD, Toyama K (1998) X Vision: A portable substrate for real-time vision applications. Comput Vis Image Und 69(1):23–37
Hamel T, Mahony R (2002) Visual servoing of an under-actuated dynamic rigid-body system: An image based approach. IEEE T Robotic Autom 18(2):187–198
Hamel T, Mahony R, Lozano R, Ostrowski J (2002) Dynamic modelling and configuration stabilization for an X4-flyer. IFAC World Congress 1(2), p 3
Hansen P, Corke PI, Boles W (2010) Wide-angle visual feature matching for outdoor localization. Int J Robot Res 29(1–2):267–297
Harris CG, Stephens MJ (1988) A combined corner and edge detector. In: Proceedings of the Fourth Alvey Vision Conference. Manchester, pp 147–151
Hart PE (2009) How the Hough transform was invented [DSP history]. IEEE Signal Proc Mag 26(6):18–22
Hartenberg RS, Denavit J (1964) Kinematic synthesis of linkages. McGraw-Hill, New York, available online at http://kmoddl.library.cornell.edu/bib.php?m=23
Hartley R, Zisserman A (2003) Multiple view geometry in computer vision. Cambridge University Press, New York
Harvey P (nd) ExifTool. http://www.sno.phy.queensu.ca/~phil/exiftool
Hashimoto K (ed) (1993) Visual servoing. In: Robotics and automated systems, vol 7. World Scientific, Singapore
Hashimoto K, Kimoto T, Ebine T, Kimura H (1991) Manipulator control with image-based visual servo. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Seoul, pp 2 267–2 272
Hellerstein JL, Diao Y, Parekh S, Tilbury DM (2004) Feedback control of computing systems. Wiley-IEEE Press, 456 p
Herschel W (1800) Experiments on the refrangibility of the invisible rays of the sun. Phil Trans R Soc Lond 90:284–292
Hill J, Park WT (1979) Real time control of a robot with a mobile camera. In: Proceedings of the 9th ISIR, SME. Washington, DC. Mar, pp 233–246
Hirata T (1996) A unified linear-time algorithm for computing distance maps. Inform Process Lett 58(3):129–133
Hirschmüller H (2008) Stereo processing by semiglobal matching and mutual information. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(2):328–341
Hirt C, Claessens S, Fecher T, Kuhn M, Pail R, Rexer M (2013) New ultrahigh-resolution picture of Earth's gravity field. Geophys Res Lett 40:4279–4283
Hoag D (1963) Consideration of Apollo IMU gimbal lock. MIT Instrumentation Laboratory, E–1344, http://www.hq.nasa.gov/alsj/e-1344.htm
Hollerbach JM (1980) A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. IEEE T Syst Man Cyb 10(11):730–736, Nov
Hollerbach JM (1982) Dynamics. In: Brady M, Hollerbach JM, Johnson TL, Lozano-Pérez T, Mason MT (eds) Robot motion – Planning and control. MIT Press, Cambridge, Massachusetts, pp 51–71
Horaud R, Canio B, Leboullenx O (1989) An analytic solution for the perspective 4-point problem. Comput Vision Graph 47(1):33–44
Horn BKP (1987) Closed-form solution of absolute orientation using unit quaternions. J Opt Soc Am A 4(4):629–642
Horn BKP, Hilden HM, Negahdaripour S (1988) Closed-form solution of absolute orientation using orthonormal matrices. J Opt Soc Am A 5(7):1 127–1 135
Hosoda K, Asada M (1994) Versatile visual servoing without knowledge of true Jacobian. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS). Munich, pp 186–193
Howard TM, Green CJ, Kelly A, Ferguson D (2008) State space sampling of feasible motions for high-performance mobile robot navigation in complex environments. J Field Robotics 25(6–7):325–345
Hu MK (1962) Visual pattern recognition by moment invariants. IRE T Inform Theor 8:179–187
Hua M-D, Ducard G, Hamel T, Mahony R, Rudin K (2014) Implementation of a nonlinear attitude estimator for aerial robotic vehicles. IEEE T Contr Syst T 22(1):201–213
Huang TS, Netravali AN (1994) Motion and structure from feature correspondences: A review. P IEEE 82(2):252–268

Humenberger M, Zinner C, Kubinger W (2009) Performance evaluation of a census-based stereo matching algorithm on embedded and multi-core hardware. In: Proceedings of the 19th International Symposium on Image and Signal Processing and Analysis (ISPA). pp 388–393

Hunt RWG (1987) The reproduction of colour, 4th ed. Fountain Press, Tolworth

Hunter RS, Harold RW (1987) The measurement of appearance. John Wiley & Sons, Inc., Chichester

Hutchinson S, Hager G, Corke PI (1996) A tutorial on visual servo control. IEEE T Robotic Autom 12(5):651–670

Iwatsuki M, Okiyama N (2002a) A new formulation of visual servoing based on cylindrical coordinate system with shiftable origin. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS). Lausanne, pp 354–359

Iwatsuki M, Okiyama N (2002b) Rotation-oriented visual servoing based on cylindrical coordinates. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Washington, DC, May, pp 4 198–4 203

Izaguirre A, Paul RP (1985) Computation of the inertial and gravitational coefficients of the dynamics equations for a robot manipulator with a load. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Mar, pp 1 024–1 032

Jägersand M, Fuentes O, Nelson R (1996) Experimental evaluation of uncalibrated visual servoing for precision manipulation. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Albuquerque, NM, pp 2 874–2 880

Jarvis RA, Byrne JC (1988) An automated guided vehicle with map building and path finding capabilities. In: Robotics Research: The Fourth international symposium. MIT Press, Cambridge, Massachusetts, pp 497–504

Jazwinski AH (2007) Stochastic processes and filtering theory. Dover Publications, Mineola

Jebara T, Azarbayejani A, Pentland A (1999) 3D structure from 2D motion. IEEE Signal Proc Mag 16(3):66–84

Julier SJ, Uhlmann JK (2004) Unscented filtering and nonlinear estimation. P IEEE 92(3):401–422

Kaehler A, Bradski G (2016) Learning OpenCV: Computer vision in C++ with the OpenCV library. O'Reilly & Associates, Köln

Kaess M, Ranganathan A, Dellaert F (2007) iSAM: Fast incremental smoothing and mapping with efficient data association. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp 1670–1677

Kahn ME (1969) The near-minimum time control of open-loop articulated kinematic linkages. Stanford University, AIM-106

Kálmán RE (1960) A new approach to linear filtering and prediction problems. J Basic Eng-T Asme 82(1):35–45

Kane TR, Levinson DA (1983) The use of Kane's dynamical equations in robotics. Int J Robot Res 2(3):3–21

Karaman S, Walter MR, Perez A, Frazzoli E, Teller S (2011) Anytime motion planning using the RRT*. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp 1478–1483

Kavraki LE, Svestka P, Latombe JC, Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE T Robotic Autom 12(4):566–580

Kelly R (1996) Robust asymptotically stable visual servoing of planar robots. IEEE T Robotic Autom 12(5):759–766

Kelly A (2013) Mobile robotics: Mathematics, models, and methods. Cambridge University Press, New York

Kelly R, Carelli R, Nasisi O, Kuchen B, Reyes F (2002a) Stable visual servoing of camera-in-hand robotic systems. IEEE-ASME T Mech 5(1):39–48

Kelly R, Shirkey P, Spong MW (2002b) Fixed-camera visual servo control for planar robots. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Washington, DC, pp 2 643–2 649

Khalil W, Creusot D (1997) SYMORO+: A system for the symbolic modelling of robots. Robotica 15(2):153–161

Khalil W, Dombre E (2002) Modeling, identification and control of robots. Kogan Page Science, London

Khatib O (1987) A unified approach for motion and force control of robot manipulators: The operational space formulation. IEEE T Robotic Autom 3(1):43–53

King-Hele D (2002) Erasmus Darwin's improved design for steering carriages and cars. Notes and Records of the Royal Society of London 56(1):41–62

Klafter RD, Chmielewski TA, Negin M (1989) Robotic engineering – An integrated approach. Prentice Hall, Upper Saddle River, New Jersey

Klein CA, Huang CH (1983) Review of pseudoinverse control for use with kinematically redundant manipulators. IEEE T Syst Man Cyb 13:245–250

Klein G, Murray D (2007) Parallel tracking and mapping for small AR workspaces. In: Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2007). pp 225–234

Klette R, Kruger N, Vaudrey T, Pauwels K, van Hulle M, Morales S, Kandil F, Haeusler R, Pugeault N, Rabe C (2011) Performance of correspondence algorithms in vision-based driver assistance using an online image sequence database. IEEE T Veh Technol 60(5):2 012–2 026

Koenderink JJ (1984) The structure of images. Biol Cybern 50(5):363–370

Koenderink JJ (2010) Color for the sciences. MIT Press, Cambridge, Massachusetts

Koenig S, Likhachev M (2002) D* Lite. In: Proceedings of the National Conference on Artificial Intelligence, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press, Cambridge, Massachusetts; 1999, pp 476–483

Koenig S, Likhachev M (2005) Fast replanning for navigation in unknown terrain. IEEE T Robotic Autom 21(3):354–363

Kriegman DJ, Hager GD, Morse AS (eds) (1998) The confluence of vision and control. Lecture notes in control and information sciences, vol 237. Springer-Verlag, Berlin Heidelberg

Kuipers JB (1999) Quaternions and rotation sequences: A primer with applications to orbits, aeroespace and virtual reality. Princeton University Press, Princeton, New Jersey

Kümmerle R, Grisetti G, Strasdat H, Konolige K, Burgard W (2011) g^2o: A general framework for graph optimization. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp 3607–3613

Lam O, Dayoub F, Schulz R, Corke P (2015) Automated topometric graph generation from floor plan analysis. In: Proceedings of the Australasian Conference on Robotics and Automation. Australasian Robotics and Automation Association (ARAA)

Lamport L (1994) LATEX: A document preparation system. User's guide and reference manual. Addison-Wesley Publishing Company, Reading

Land EH, McCann J (1971) Lightness and retinex theory. J Opt Soc Am A 61(1):1–11

Land MF, Nilsson D-E (2002) Animal eyes. Oxford University Press, Oxford

LaValle SM (1998) Rapidly-exploring random trees: A new tool for path planning. Computer Science Dept., Iowa State University, TR 98–11

LaValle SM (2006) Planning algorithms. Cambridge University Press, New York

LaValle SM (2011a) Motion planning: The essentials. IEEE Robot Autom Mag 18(1):79–89

LaValle SM (2011b) Motion planning: Wild frontiers. IEEE Robot Autom Mag 18(2):108–118

LaValle SM, Kuffner JJ (2001) Randomized kinodynamic planning. Int J Robot Res 20(5):378–400

Laussedat A (1899) La métrophotographie. Enseignement supérieur de la photographie. Gauthier-Villars, 52 p

Leavers VF (1993) Which Hough transform? Comput Vis Image Und 58(2):250–264

Lee CSG, Lee BH, Nigham R (1983) Development of the generalized D'Alembert equations of motion for mechanical manipulators. In: Proceedings of the 22nd CDC, San Antonio, Texas. pp 1 205–1 210

Lepetit V, Moreno-Noguer F, Fua P (2009) EPnP: An accurate $O(n)$ solution to the PnP problem. Int J Comput Vision 81(2):155–166

Li H, Hartley R (2006) Five-point motion estimation made easy. In: 18th International Conference on Pattern Recognition ICPR 2006. Hong Kong, pp 630–633

Li Y, Jia W, Shen C, van den Hengel A (2014) Characterness: An indicator of text in the wild. IEEE T Image Process 23(4):1666–1677

Li T, Bolic M, Djuric P (2015) Resampling methods for particle filtering: Classification, implementation, and strategies. IEEE Signal Proc Mag 32(3):70–86

Lin Z, Zeman V, Patel RV (1989) On-line robot trajectory planning for catching a moving object. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp 1 726–1 731

Lindeberg T (1993) Scale-space theory in computer vision. Springer-Verlag, Berlin Heidelberg

Lloyd J, Hayward V (1991) Real-time trajectory generation using blend functions. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Seoul, pp 784–789

Longuet-Higgins H (1981) A computer algorithm for reconstruction of a scene from two projections. Nature 293:133–135

Lovell J, Kluger J (1994) Apollo 13. Coronet Books

Lowe DG (1991) Fitting parametrized three-dimensional models to images. IEEE T Pattern Anal 13(5): 441–450

Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vision 60(2):91–110

Lowry S, Sunderhauf N, Newman P, Leonard J, Cox D, Corke P, Milford M (2015) Visual place recognition: A survey. Robotics, IEEE Transactions on (99):1–19

Lu F, Milios E (1997) Globally consistent range scan alignment for environment mapping. Auton Robot 4:333–349

Lucas SM (2005) ICDAR 2005 text locating competition results. In: Proceedings of the Eighth International Conference on Document Analysis and Recognition, ICDAR05. pp 80–84

Lucas BD, Kanade T (1981) An iterative image registration technique with an application to stereo vision. In: International joint conference on artificial intelligence (IJCAI), Vancouver, vol 2. http://ijcai.org/Past%20Proceedings/IJCAI-81-VOL-2/PDF/017.pdf, pp 674–679

Luh JYS, Walker MW, Paul RPC (1980) On-line computational scheme for mechanical manipulators. J Dyn Syst-T ASME 102(2):69–76

Lumelsky V, Stepanov A (1986) Dynamic path planning for a mobile automaton with limited information on the environment. IEEE T Automat Contr 31(11):1 058–1 063

Luong QT (1992) matrice fondamentale et autocalibration en vision par ordinateur. Ph.D. thesis, Université de Paris-Sud, Orsay, France

Lynch KM, Park FC (2017) Modern robotics: Mechanics, planning, and control. Cambridge University Press, New York

Ma Y, Kosecka J, Soatto S, Sastry S (2003) An invitation to 3D. Springer-Verlag, Berlin Heidelberg

Magnusson M, Lilienthal A, Duckett T (2007) Scan registration for autonomous mining vehicles using 3D-NDT. J Field Robotics 24(10):803–827

Magnusson M, Nuchter A, Lorken C, Lilienthal AJ, Hertzberg J (2009) Evaluation of 3D registration reliability and speed – A comparison of ICP and NDT. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp 3907–3912

Mahony R, Kumar V, Corke P (2012) Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. IEEE Robot Autom Mag (19):20–32

Maimone M, Cheng Y, Matthies L (2007) Two years of visual odometry on the Mars exploration rovers. J Field Robotics 24(3):169–186

Makhlin AG (1985) Stability and sensitivity of servo vision systems. In: Proc 5th International Conference on Robot Vision and Sensory Controls – RoViSeC 5. IFS (Publications), Amsterdam, pp 79–89

Malis E (2004) Improving vision-based control using efficient second-order minimization techniques. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp 1843–1848

Malis E, Vargas M (2007) Deeper understanding of the homography decomposition for vision-based control. Research Report, RR-6303, Institut National de Recherche en Informatique et en Automatique (INRIA), 90 p, https://hal.inria.fr/inria-00174036v3/document

Malis E, Chaumette F, Boudet S (1999) 2-1/2D visual servoing. IEEE T Robotic Autom 15(2):238–250

Marey M, Chaumette F (2008) Analysis of classical and new visual servoing control laws. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Pasadena, pp 3244–3249

Mariottini GL, Prattichizzo D (2005) EGT for multiple view geometry and visual servoing: Robotics vision with pinhole and panoramic cameras. IEEE T Robotic Autom 12(4):26–39

Mariottini GL, Oriolo G, Prattichizzo D (2007) Image-based visual servoing for nonholonomic mobile robots using epipolar geometry. IEEE T Robotic Autom 23(1):87–100

Marr D (2010) Vision: A computational investigation into the human representation and processing of visual information. MIT Press, Cambridge, Massachusetts

Martin D, Fowlkes C, Tal D, Malik J (2001) A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. Proceedings of the 8th International Conference on Computer Vision, vol 2. pp 416–423

Martins FN, Celeste WC, Carelli R, Sarcinelli-Filho M, Bastos-Filho TF (2008) An adaptive dynamic controller for autonomous mobile robot trajectory tracking. Control Eng Pract 16(11):1354–1363

Masutani Y, Mikawa M, Maru N, Miyazaki F (1994) Visual servoing for non-holonomic mobile robots. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS). Munich, pp 1 133–1 140

Matarić MJ (2007) The robotics primer. MIT Press, Cambridge, Massachusetts

Matas J, Chum O, Urban M, Pajdla T (2004) Robust wide-baseline stereo from maximally stable extremal regions. Image Vision Comput 22(10):761–767

Matthews ND, An PE, Harris CJ (1995) Vehicle detection and recognition for autonomous intelligent cruise control. Technical Report, University of Southampton

Matthies L (1992) Stereo vision for planetary rovers: Stochastic modeling to near real-time implementation. Int J Comput Vision 8(1):71–91

Mayeda H, Yoshida K, Osuka K (1990) Base parameters of manipulator dynamic models. IEEE T Robotic Autom 6(3):312–321

McLauchlan PF (1999) The variable state dimension filter applied to surface-based structure from motion. University of Surrey, VSSP-TR-4/99

Merlet JP (2006) Parallel robots. Kluwer Academic, Dordrecht

Mettler B (2003) Identification modeling and characteristics of miniature rotorcraft. Kluwer Academic, Dordrecht

Mičušík B, Pajdla T (2003) Estimation of omnidirectional camera model from epipolar geometry. In: IEEE Conference on Computer Vision and Pattern Recognition, vol 1. Madison, pp 485–490

Middleton RH, Goodwin GC (1988) Adaptive computed torque control for rigid link manipulations. Syst Control Lett 10(1):9–16

Mikolajczyk K, Schmid C (2004) Scale and affine invariant interest point detectors. Int J Comput Vision 60(1):63–86

Mikolajczyk K, Schmid C (2005) A performance evaluation of local descriptors. IEEE T Pattern Anal 27(10):1 615–1 630

Mindell DA (2008) Digital Apollo. MIT Press, Cambridge, Massachusetts

Molton N, Brady M (2000) Practical structure and motion from stereo when motion is unconstrained. Int J Comput Vision 39(1):5–23

Montemerlo M, Thrun S (2007) FastSLAM: A scalable method for the simultaneous localization and mapping problem in robotics, vol 27. Springer-Verlag, Berlin Heidelberg
Montemerlo M, Thrun S, Koller D, Wegbreit B (2002) FastSLAM: A factored solution to the simultaneous localization and mapping problem. In: Proceedings of the AAAI National Conference on Artificial Intelligence. AAAI, Edmonton, Canada
Montemerlo M, Thrun S, Koller D, Wegbreit B (2003) FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence. Morgan Kaufmann, San Francisco, pp 1151–1156
Moravec H (1980) Obstacle avoidance and navigation in the real world by a seeing robot rover. Ph.D. thesis, Stanford University
Morel G, Liebezeit T, Szewczyk J, Boudet S, Pot J (2000) Explicit incorporation of 2D constraints in vision based control of robot manipulators. In: Corke PI, Trevelyan J (eds) Lecture notes in control and information sciences. Experimental robotics VI, vol 250. Springer-Verlag, Berlin Heidelberg, pp 99–108
Muja M, Lowe DG (2009) Fast approximate nearest neighbors with automatic algorithm configuration. International Conference on Computer Vision Theory and Applications (VISAPP), Lisbon, Portugal (Feb 2009), pp 331–340
Murray RM, Sastry SS, Zexiang L (1994) A mathematical introduction to robotic manipulation. CRC Press, Inc., Boca Raton
NASA (1970) Apollo 13: Technical air-to-ground voice transcription. Test Division, Apollo Spacecraft Program Office, http://www.hq.nasa.gov/alsj/a13/AS13_TEC.PDF
Nayar SK (1997) Catadioptric omnidirectional camera. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Los Alamitos, CA, pp 482–488
Neilson S (2011) Robot nation: Surviving the greatest socio-economic upheaval of all time. Eridanus Press, New York, 124 p
Neira J, Tardós JD (2001) Data association in stochastic mapping using the joint compatibility test. IEEE T Robotic Autom 17(6):890–897
Neira J, Davison A, Leonard J (2008) Guest editorial special issue on Visual SLAM. IEEE T Robotic Autom 24(5):929–931
Nethery JF, Spong MW (1994) Robotica: A mathematica package for robot analysis. IEEE T Robotic Autom 1(1):13–20
Newcombe RA, Lovegrove SJ, Davison AJ (2011) DTAM: Dense tracking and mapping in real-time. In: Proceedings of the International Conference on Computer Vision, pp 2320–2327
Newman P (n.d.) C4B mobile robots and estimation resources. Oxford University. http://www.robots.ox.ac.uk/~pnewman/Teaching/C4CourseResources/C4BResources.html
Ng J, Bräunl T (2007) Performance comparison of bug navigation algorithms. J Intell Robot Syst 50(1):73–84
Niblack W (1985) An introduction to digital image processing. Strandberg Publishing Company Birkeroed, Denmark
Nilsson NJ (1971) Problem-solving methods in artificial intelligence. McGraw-Hill, New York
Nistér D (2003) An efficient solution to the five-point relative pose problem. In: IEEE Conference on Computer Vision and Pattern Recognition, vol 2. Madison, pp 195–202
Nistér D, Naroditsky O, Bergen J (2006) Visual odometry for ground vehicle applications. J Field Robotics 23(1):3–20
Nixon MS, Aguado AS (2012) Feature extraction and image processing, 3rd ed. Academic Press, London Oxford
Noble JA (1988) Finding corners. Image Vision Comput 6(2):121–128
Okutomi M, Kanade T (1993) A multiple-baseline stereo. IEEE T Pattern Anal 15(4):353–363
Ollis M, Herman H, Singh S (1999) Analysis and design of panoramic stereo vision using equi-angular pixel cameras. Robotics Institute, Carnegie Mellon University, CMU-RI-TR-99-04, Pittsburgh, PA
Olson E (2011) AprilTag: A robust and flexible visual fiducial system. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). pp 3400–3407
Orin DE, McGhee RB, Vukobratovic M, Hartoch G (1979) Kinematics and kinetic analysis of open-chain linkages utilizing Newton-Euler methods. Math Biosci 43(1/2):107–130
Ortega R, Spong MW (1989) Adaptive motion control of rigid robots: A tutorial. Automatica 25(6):877–888
Otsu N (1975) A threshold selection method from gray-level histograms. Automatica 11:285–296
Papanikolopoulos NP, Khosla PK (1993) Adaptive robot visual tracking: Theory and experiments. IEEE T Automat Contr 38(3):429–445
Papanikolopoulos NP, Khosla PK, Kanade T (1993) Visual tracking of a moving target by a camera mounted on a robot: A combination of vision and control. IEEE T Robotic Autom 9(1):14–35
Park FC (1994) Computational aspects of the product-of-exponentials formula for robot kinematics. IEEE T Automat Contr 39(3):643–647
Paul R (1972) Modelling, trajectory calculation and servoing of a computer controlled arm. Ph.D. thesis, technical report AIM-177, Stanford University

Paul R (1979) Manipulator Cartesian path control. IEEE T Syst Man Cyb 9:702–711

Paul RP (1981) Robot manipulators: Mathematics, programming, and control. MIT Press, Cambridge, Massachusetts

Paul RP, Shimano B (1978) Kinematic control equations for simple manipulators. In: IEEE Conference on Decision and Control, vol 17. pp 1 398–1 406

Paul RP, Zhang H (1986) Computationally efficient kinematics for manipulators with spherical wrists based on the homogeneous transformation representation. Int J Robot Res 5(2):32–44

Piepmeier JA, McMurray G, Lipkin H (1999) A dynamic quasi-Newton method for uncalibrated visual servoing. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Detroit, pp 1 595–1 600

Pilu M (1997) A direct method for stereo correspondence based on singular value decomposition. In: Proceedings of the Computer Vision and Pattern Recognition, IEEE Computer Society, San Juan, pp 261–266

Pivtoraiko M, Knepper RA, Kelly A (2009) Differentially constrained mobile robot motion planning in state lattices. J Field Robotics 26(3):308–333

Pock T (2008) Fast total variation for computer vision. Ph.D. thesis, Graz University of Technology

Pollefeys M, Nistér D, Frahm JM, Akbarzadeh A, Mordohai P, Clipp B, Engels C, Gallup D, Kim SJ, Merrell P, et al. (2008) Detailed real-time urban 3D reconstruction from video. Int J Comput Vision 78(2):143–167, Jul

Pomerleau D, Jochem T (1995) No hands across America Journal. http://www.cs.cmu.edu/~tjochem/nhaa/Journal.html

Pomerleau D, Jochem T (1996) Rapidly adapting machine vision for automated vehicle steering. IEEE Expert 11(1):19–27

Posner I, Corke P, Newman P (2010) Using text-spotting to query the world. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp 3181–3186

Pounds P (2007) Design, construction and control of a large quadrotor micro air vehicle. Ph.D. thesis, Australian National University

Pounds P, Mahony R, Gresham J, Corke PI, Roberts J (2004) Towards dynamically-favourable quad-rotor aerial robots. In: Proceedings of the Australasian Conference on Robotics and Automation. Canberra

Pounds P, Mahony R, Corke PI (2006) A practical quad-rotor robot. In: Proceedings of the Australasian Conference on Robotics and Automation. Auckland

Pounds P, Mahony R, Corke PI (2007) System identification and control of an aerobot drive system. In: Information, Decision and Control. IEEE, pp 154–159

Poynton CA (2003) Digital video and HDTV: Algorithms and interfaces. Morgan Kaufmann, San Francisco

Poynton CA (2012) Digital video and HD algorithms and interfaces. Morgan Kaufmann, Burlington

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2007) Numerical recipes, 3rd ed. Cambridge University Press, New York

Prince SJ (2012) Computer vision: Models, learning, and inference. Cambridge University Press, New York

Prouty RW (2002) Helicopter performance, stability, and control. Krieger, Malabar FL

Pynchon T (2006) Against the day. Jonathan Cape, London

Rekleitis IM (2004) A particle filter tutorial for mobile robot localization. Technical report (TR-CIM-04-02), Centre for Intelligent Machines, McGill University

Rives P, Chaumette F, Espiau B (1989) Positioning of a robot with respect to an object, tracking it and estimating its velocity by visual servoing. In: Hayward V, Khatib O (eds) Lecture notes in control and information sciences. Experimental robotics I, vol 139. Springer-Verlag, Berlin Heidelberg, pp 412–428

Rizzi AA, Koditschek DE (1991) Preliminary experiments in spatial robot juggling. In: Chatila R, Hirzinger G (eds) Lecture notes in control and information sciences. Experimental robotics II, vol 190. Springer-Verlag, Berlin Heidelberg, pp 282–298

Roberts LG (1963) Machine perception of three-dimensional solids. MIT Lincoln Laboratory, TR 315, http://www.packet.cc/files/mach-per-3D-solids.html

Rosenfield GH (1959) The problem of exterior orientation in photogrammetry. Photogramm Eng 25(4):536–553

Rosten E, Porter R, Drummond T (2010) FASTER and better: A machine learning approach to corner detection. IEEE T Pattern Anal 32:105–119

Russell S, Norvig P (2009) Artificial intelligence: A modern approach, 3rd ed. Prentice Hall Press, Upper Saddle River, NJ

Sakaguchi T, Fujita M, Watanabe H, Miyazaki F (1993) Motion planning and control for a robot performer. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Atlanta, May, pp 925–931

Salvi J, Matabosch C, Fofi D, Forest J (2007) A review of recent range image registration methods with accuracy evaluation. Image Vision Comput 25(5):578–596

Samson C, Espiau B, Le Borgne M (1990) Robot control: The task function approach. Oxford University Press, Oxford

Sanderson AC, Weiss LE, Neuman CP (1987) Dynamic sensor-based control of robots with visual feedback. IEEE T Robotic Autom RA-3(5):404–417

Scaramuzza D, Fraundorfer F (2011) Visual odometry [tutorial]. IEEE Robot Autom Mag 18(4):80–92

Scharstein D, Pal C (2007) Learning conditional random fields for stereo. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007). Minneapolis, MN

Scharstein D, Szeliski R (2002) A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int J Comput Vision 47(1):7–42

Selig JM (2005) Goemetric fundamentals of robotics. Springer-Verlag, Berlin Heidelberg

Sharp A (1896) Bicycles & tricycles: An elementary treatise on their design an construction; With examples and tables. Longmans, Green and Co., London New York Bombay

Sheridan TB (2003) Telerobotics, automation, and human supervisory control. MIT Press, Cambridge, Massachusetts, 415 p

Shi J, Tomasi C (1994) Good features to track. In: Proceedings of the Computer Vision and Pattern Recognition. IEEE Computer Society, Seattle, pp 593–593

Shih FY (2009) Image processing and mathematical morphology: Fundamentals and applications, CRC Press, Boca Raton

Shirai Y (1987) Three-dimensional computer vision. Springer-Verlag, New York

Shirai Y, Inoue H (1973) Guiding a robot by visual feedback in assembling tasks. Pattern Recogn 5(2):99–106

Shoemake K (1985) Animating rotation with quaternion curves. In: Proceedings of ACM SIGGRAPH, San Francisco, pp 245–254

Siciliano B, Khatib O (eds) (2016) Springer handbook of robotics, 2nd ed. Springer-Verlag, New York

Siciliano B, Sciavicco L, Villani L, Oriolo G (2009) Robotics: Modelling, planning and control. Springer-Verlag, Berlin Heidelberg

Siegwart R, Nourbakhsh IR, Scaramuzza D (2011) Introduction to autonomous mobile robots, 2nd ed. MIT Press, Cambridge, Massachusetts

Silver WM (1982) On the equivalance of Lagrangian and Newton-Euler dynamics for manipulators. Int J Robot Res 1(2):60–70

Sivic J, Zisserman A (2003) Video Google: A text retrieval approach to object matching in videos. In: Proceedings of the Ninth IEEE International Conference on Computer Vision. pp 1 470–1 477

Skaar SB, Brockman WH, Hanson R (1987) Camera-space manipulation. Int J Robot Res 6(4):20–32

Skofteland G, Hirzinger G (1991) Computing position and orientation of a freeflying polyhedron from 3D data. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Seoul, pp 150–155

Slama CC (ed) (1980) Manual of photogrammetry, 4th ed. American Society of Photogrammetry

Smith R (2007) An overview of the Tesseract OCR engine. In: 9th International Conference on Document Analysis and Recognition (ICDAR). pp 629–633

Sobel D (1996) Longitude: The true story of a lone genius who solved the greatest scientific problem of his time. Fourth Estate, London

Soille P (2003) Morphological image analysis: Principles and applications. Springer-Verlag, Berlin Heidelberg

Spong MW (1989) Adaptive control of flexible joint manipulators. Syst Control Lett 13(1):15–21

Spong MW, Hutchinson S, Vidyasagar M (2006) Robot modeling and control, 2nd ed. John Wiley & Sons, Inc., Chichester

Srinivasan VV, Venkatesh S (1997) From living eyes to seeing machines. Oxford University Press, Oxford

Stachniss C, Burgard W (2014) Particle filters for robot navigation. Foundations and Trends in Robotics 3(4):211–282

Steinvall A (2002) English colour terms in context. Ph.D. thesis, Ume Universitet

Stentz A (1994) The D* algorithm for real-time planning of optimal traverses. The Robotics Institute, Carnegie-Mellon University, CMU-RI-TR-94-37

Stewart A (2014) Localisation using the appearance of prior structure. Ph.D. thesis, University of Oxford

Stone JV (2012) Vision and brain: How we perceive the world. MIT Press, Cambridge, Massachusetts

Strasdat H (2012) Local accuracy and global consistency for efficient visual SLAM. Ph.D. thesis, Imperial College London

Strelow D, Singh S (2004) Motion estimation from image and inertial measurements. Int J Robot Res 23(12):1 157–1 195

Sünderhauf N (2012) Robust optimization for simultaneous localization and mapping. Ph.D. thesis, Technische Universität Chemnitz

Sussman GJ, Wisdom J, Mayer ME (2001) Structure and interpretation of classical mechanics. MIT Press, Cambridge, Massachusetts

Sutherland IE (1974) Three-dimensional data input by tablet. P IEEE 62(4):453–461

Svoboda T, Pajdla T (2002) Epipolar geometry for central catadioptric cameras. Int J Comput Vision 49(1):23–37

Szeliski R (2011) Computer vision: Algorithms and applications. Springer-Verlag, Berlin Heidelberg

Tahri O, Chaumette F (2005) Point-based and region-based image moments for visual servoing of planar objects. IEEE T Robotic Autom 21(6):1 116–1 127

Tahri O, Mezouar Y, Chaumette F, Corke PI (2009) Generic decoupled image-based visual servoing for cameras obeying the unified projection model. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Kobe, pp 1 116–1 121

Taylor RA (1979) Planning and execution of straight line manipulator trajectories. IBM J Res Dev 23(4):424–436

ter Haar Romeny BM (1996) Introduction to scale-space theory: Multiscale geometric image analysis. Utrecht University

Thrun S, Burgard W, Fox D (2005) Probabilistic robotics. MIT Press, Cambridge, Massachusetts

Tissainayagam P, Suter D (2004) Assessing the performance of corner detectors for point feature tracking applications. Image Vision Comput 22(8):663–679

Titterton DH, Weston JL (2005) Strapdown inertial navigation technology. IEE Radar, Sonar, Navigation and Avionics Series, vol 17, The Institution of Engineering and Technology (IET), 576 p

Tomasi C, Kanade T (1991) Detection and tracking of point features. Carnegie Mellon University, CMU-CS-91-132

Triggs B, McLauchlan P, Hartley R, Fitzgibbon A (2000) Bundle adjustment – A modern synthesis. Lecture notes in computer science. Vision algorithms: theory and practice, vol 1 883. Springer-Verlag, Berlin Heidelberg, pp 153–177

Tsakiris D, Rives P, Samson C (1998) Extending visual servoing techniques to nonholonomic mobile robots. In: Kriegman DJ, Hager GD, Morse AS (eds) Lecture notes in control and information sciences. The confluence of vision and control, vol 237. Springer-Verlag, Berlin Heidelberg, pp 106–117

Uicker JJ (1965) On the dynamic analysis of spatial linkages using 4 by 4 matrices. Dept. Mechanical Engineering and Astronautical Sciences, NorthWestern University

Usher K (2005) Visual homing for a car-like vehicle. Ph.D. thesis, Queensland University of Technology

Usher K, Ridley P, Corke PI (2003) Visual servoing of a car-like vehicle – An application of omnidirectional vision. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Taipai, Sep, pp 4 288–4 293

Valgren C, Lilienthal AJ (2010) SIFT, SURF & seasons: Appearance-based long-term localization in outdoor environments. Robot Auton Syst 58(2):149–156

Vanderborght B, Sugar T, Lefeber D (2008) Adaptable compliance or variable stiffness for robotic applications. IEEE Robot Autom Mag 15(3):8–9

Vedaldi A, Fulkerson B (2008) VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org

Wade NJ (2007) Image, eye, and retina. J Opt Soc Am A 24(5):1229–1249

Walker MW, Orin DE (1982) Efficient dynamic computer simulation of robotic mechanisms. J Dyn Syst-T ASME 104(3):205–211

Walter WG (1950) An imitation of life. Sci Am 182(5):42–45

Walter WG (1951) A machine that learns. Sci Am 185(2):60–63

Walter WG (1953) The living brain. Duckworth, London

Warren M (2015) Long-range stereo visual odometry for unmanned aerial vehicles. Ph.D. thesis, Queensland University of Technology

Weiss LE (1984) Dynamic visual servo control of robots: An adaptive image-based approach. Ph.D. thesis, technical report CMU-RI-TR-84-16, Carnegie-Mellon University

Weiss L, Sanderson AC, Neuman CP (1987) Dynamic sensor-based control of robots with visual feedback. IEEE T Robotic Autom 3(1):404–417

Westmore DB, Wilson WJ (1991) Direct dynamic control of a robot using an end-point mounted camera and Kalman filter position estimation. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Seoul, Apr, pp 2 376–2 384

Whitney DE (1969) Resolved motion rate control of manipulators and human prostheses. IEEE T Man Machine 10(2):47–53

Wiener N (1965) Cybernetics or control and communication in the animal and the machine. MIT Press, Cambridge, Massachusetts

Wilburn B, Joshi N, Vaish V, Talvala E-V, Antunez E, Barth A, Adams A, Horowitz M, Levoy M (2005) High performance imaging using large camera arrays. ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2005 24(3):765–776

Wolf PR (1974) Elements of photogrammetry. McGraw-Hill, New York

Woodfill J, Von Herzen B (1997) Real-time stereo vision on the PARTS reconfigurable computer. In: Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines, Grenoble. pp 201–210

Xu G, Zhang Z (1996) Epipolar geometry in stereo, motion, and object recognition: A unified approach. Springer-Verlag, Berlin Heidelberg

Ying X, Hu Z (2004) Can we consider central catiodioptric cameras and fisheye cameras within a unified imaging model. In: Pajdla T, Matas J (eds) Lecture notes in computer science. Computer vision – ECCV 2004, vol 3 021. Springer-Verlag, Berlin Heidelberg, pp 442–455

Yoshikawa T (1984) Analysis and control of robot manipulators with redundancy. In: Brady M, Paul R (eds) Robotics research: The first international symposium. MIT Press, Cambridge, Massachusetts, pp 735–747

Zabih R, Woodfill J (1994) Non-parametric local transforms for computing visual correspondence. In: Ecklundh J-O (ed) Lecture notes in computer science. Computer Vision – ECCV 1994, vol 800. Springer-Verlag, Berlin Heidelberg, pp 151–158

Zarchan P, Musoff H (2005) Fundamentals of Kalman filtering: A practical approach. Progress in Astronautics and Aeronautics, vol 208. American Institute of Aeronautics and Astronautics

Zhang Z, Faugeras O, Kohonen T, Hunag TS, Schroeder MR (1992) Three D-dynamic scene analysis: A stereo based approach. Springer-Verlag, New York

Ziegler J, Bender P, Schreiber M, Lategahn H, Strauss T, Stiller C, Thao Dang, Franke U, Appenrodt N, Keller CG, Kaus E, Herrtwich RG, Rabe C, Pfeiffer D, Lindner F, Stein F, Erbs F, Enzweiler M, Knöppel C, Hipp J, Haueis M, Trepte M, Brenk C, Tamke A, Ghanaat M, Braun M, Joos A, Fritz H, Mock H, Hein M, Zeeb E (2014) Making Bertha drive – An autonomous journey on a historic route. IEEE Intelligent Transportation Systems Magazine 6(2):8–20

Index

Index of People

Index of Functions, Classes and Methods

Classes are shown in **bold**, Simulink® models in *italics*, and methods are prefixed by a dot. All others are Toolbox functions.

General Index