Introduction to Mobile Robotics

Jeff McGough
Larry Pyeatt

Department of Computer Science
South Dakota School of Mines and Technology
Rapid City, SD 57701, USA

September 5, 2012

Forward position kinematics

The forward position kinematics (FPK) solves the following problem: "Given the joint positions, what is the corresponding end effector's pose?"

Serial chains

The solution is always unique: one given joint position vector always corresponds to only one single end effector pose. The FK problem is not difficult to solve, even for a completely arbitrary kinematic structure.

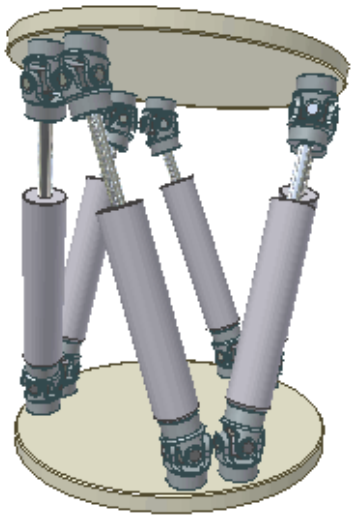
Methods for a forward kinematic analysis:

- using straightforward geometry
- using transformation matrices

Forward position kinematics

Parallel chains (Stewart- Gough Manipulators)

The solution is not unique: one set of joint coordinates has more different end effector poses. In case of a Stewart platform there are 40 poses possible which can be real for some design examples. Computation is intensive but solved in closed form with the help of algebraic geometry.



Forward velocity kinematics

Forward velocity kinematics

The forward velocity kinematics (FVK) solves the following problem: "Given the vectors of joint positions and joint velocities, what is the resulting end effector twist?" The solution is always unique: one given set of joint positions and joint velocities always corresponds to only one single end effector twist.

Inverse kinematics

Inverse position kinematics

The inverse position kinematics (IPK) solves the following problem: "Given the actual end effector pose, what are the corresponding joint positions?" In contrast to the forward problem, the solution of the inverse problem is not always unique: the same end effector pose can be reached in several configurations, corresponding to distinct joint position vectors.

A 6R manipulator (a serial chain with six revolute joints) with a completely general geometric structure has sixteen different inverse kinematics solutions, found as the solutions of a sixteenth order polynomial.

Inverse velocity kinematics

Inverse velocity kinematics

Assuming that the inverse position kinematics problem has been solved for the current end effector pose, the inverse velocity kinematics (IVK) then solves the following problem: "Given the end effector twist, what is the corresponding vector of joint velocities?"

Force kinematics

Forward force kinematics

The forward force kinematics (FFK) solves the following problem: "Given the vectors of joint force/torques, what is the resulting static wrench that the end effector exerts on the environment?" (If the end effector is rigidly fixed to a rigid environment.)

Inverse force kinematics

Assuming that the inverse position kinematics problem has been solved for the current end effector pose, the inverse force kinematics (IFK) then solves the following problem: "Given the wrench that acts on the end effector, what is the corresponding vector of joint forces/torques?"

Math Elements

- Points
- Vectors
- Matrices
- Norms
- Unit vector
- Dot product
- Orthogonal
- Projection

Mathematical Style

- Real line: $\mathbb{R}$
- Plane: $\mathbb{R}^2$
- Space: $\mathbb{R}^3$
- Workspace: $\mathcal{W}$
- Obstacles: $\mathcal{WO}_i$
- Free space: $\mathcal{W} \setminus \bigcup_i \mathcal{WO}_i$
- Point in space: $x = (x_1, x_2, x_3)$
- Vector: $\vec{v} \in \mathbb{R}^n$ or more often $v = [v_1, v_2, \ldots, v_n]^T$.
- Bounded workspace: $\mathcal{W} \subset B_r(x) \equiv \{y \in \mathbb{R}^n | d(x, y) < r\}$ for some $0 < r < \infty$.

Mathematical Style

Vectors

$x \in \mathbb{R}^n$: $x = \{x_1, x_2, ...x_n\}$, often this is written as $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$.

Let $c \in \mathbb{R}$ and $x, y \in \mathbb{R}^n$ then

- $x + y = \{x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n\}$
- $cx = \{cx_1, cx_2, \ldots, cx_n\}$
- $x \cdot y = \sum_{i=1}^{n} x_i y_i$
- $\|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$

Matrices

Let $A, B \in \mathbb{R}^{n \times n}$, $A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$, $B = \begin{pmatrix} b_{11} & \dots & b_{1n} \\ \dots & \dots & \dots \\ b_{n1} & \dots & b_{nn} \end{pmatrix}$

- $A + B = \begin{pmatrix} a_{11} + b_{11} & \dots & a_{1n} + b_{1n} \\ \dots & \dots & \dots \\ a_{n1} + b_{n1} & \dots & a_{nn} + b_{nn} \end{pmatrix}$

- $AB = \begin{pmatrix} c_{11} & \dots & c_{1n} \\ \dots & \dots & \dots \\ c_{n1} & \dots & c_{nn} \end{pmatrix}$, $c_{ij} = \sum_k a_{ik} b_{kj}$

Mathematics Review

- $Ax = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{n1} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \sum_k a_{1k} x_k \\ \sum_k a_{2k} x_k \\ \vdots \\ \sum_k a_{nk} x_k \end{pmatrix}$

- $I = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix}$

Mathematics Review

Matrices ....

- Transpose: A^T: $\{a_{ij}\}^T = \{a_{ji}\}$
- Determinant: $\det(A)$
- $(AB)^T = B^T A^T$
- $\det(AB) = \det(A)\det(B)$
- Symmetric: $A^T = A$
- Symmetric Positive Definite: $x^T Ax > 0$ for $x \neq 0$.

Note that by the definition $AB = O(n^3)$.

Strassen's algorithm $O(n^{2.807})$ and Coppersmith-Winograd $O(n^{2.376})$.

Mathematics Review

Linear Systems

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n &= b_n \end{aligned} \quad \Rightarrow \quad Ax = b$$

One approach is to solve via Gauss Elimination (ok).

The industry approach - LU factorization:

1. $A \rightarrow LU$, $O(n^3)$
2. Solve $Lc = b \rightarrow c$, $O(n^2)$
3. Solve $Ux = c \rightarrow x$, $O(n^2)$

You don't actually have to know how to do this, just to call the solvers.

Inverses

The inverse of A is notated A^{-1}:

$$A(A^{-1}) = I = (A^{-1})A$$

Given the inverse:

$$Ax = b \rightarrow x = A^{-1}b$$

Is this a good approach to solving $Ax = b$? No.

The fast multiplication algorithms are not numerically stable. Best to use LU. LU can also make good use of matrix structure.

Inverses

Possible that an algorithm may involve computing an inverse, but this can often be converted to a linear solve:

$$y^* = y + BC^{-1}x$$

Introduce a new vector z, s.t. $Cz = x$:

$$y^* = y + BC^{-1}Cz$$

Since $C^{-1}C = I$, we only have to solve $y^* = y + Bz$ to find z, then solve $Cz = x$ to find x. So, we do two simple solves that do not involve finding an inverse (use LU decomposition).

Numerics

GSL is an open source C numerics library.

- Special functions
- Vectors and Matrices
- Permuations, combinations and sorting
- BLAS
- Linear Algebra
- FFTs
- Integration and differentiation
- Random number and distributions
- Statistics
- ODEs
- Interpolation and least squares
- Roots and minima

GSL Details

- Must have `#include <gsl/gsllib.h>`
- Compile: `gcc -Wall -I/usr/local/include -c example.c`
- Link: `gcc -L/usr/local/lib example.o -lgsl -lgslcblas -lm -o example`
- Speedup: `-DHAVE_INLINE` in the compile line
- C++ works
- See section 2.2 and 2.3 in the GSL manual for more details.

GSL Details

```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <gsl/gsl_rng.h>
#include <gsl/gsl_randist.h>

#define LEN 20


int main()
{
  int i;
  double x, y, z;
  const gsl_rng_type * T;
  gsl_rng * r;
```

GSL Details

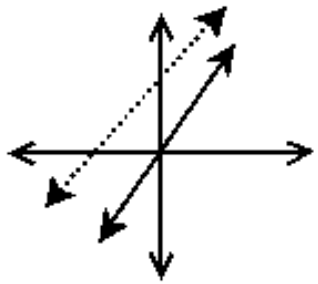
```
/* create a generator chosen by the
    environment variable GSL_RNG_TYPE */

 T = gsl_rng_default;
 r = gsl_rng_alloc (T);
 gsl_rng_set(r, time(NULL));

 for(i=0;i<LEN;i++)
   {
     x = 1.0+ gsl_ran_gaussian (r, 0.1);
     y = 2.0+ gsl_ran_gaussian (r, 0.2);
     z = x + 0.1*y;
     printf("%f\n",z);
   }
 return 0;
}
```

GSL Details

- Allocate a vector:
  `gsl_vector *vector = gsl_vector_alloc (n);`
- Allocate a matrix:
  `gsl_matrix *matrix = gsl_matrix_alloc (n , n);`
- Allocate a permutation vector:
  `gsl_permutation p = gsl_permutation_alloc (n);`
- Set a value in a vector:
  `gsl_vector_set(vector, i_location, value);`
- Set a value in a matrix:
  `gsl_matrix_set(vector, i_location, j_location, value);`
- Multiply matrix times vector:
  `gsl_blas_dgemv(CblasNoTrans,1,matrix,vector,0,answer);`
- Form the LU factorization:
  `gsl_linalg_LU_decomp (matrix, permutation, &s);`
- Solve a linear system:
  `gsl_linalg_LU_solve (matrix, permuation, right_hand_side, answer);`

Terms

Vector Space V: if $c \in \mathbb{R}$ and $x, y \in V$ then $x + y \in V$ and $cx \in V$.
One example is a line in the plane (solid is, dotted is not):



Note that $0 \in V$. We can represent the line by a vector

$$t \begin{pmatrix} a \\ b \end{pmatrix}$$

where $t \in \mathbb{R}$, t is the scale factor. Another example is the collection of all 2×2 matrices:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

Mathematics Review

Subspace: subset of V that is a vector space as well.
Span of

$$\left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right\}$$

The subspace of 2×2 matrices: is made from the span of

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

The sets of elements for which you can generate the subspace are called a basis.

Mathematics Review

Nullspace: subspace, $\mathcal{N}(A)$, all w such that $Aw = 0$.
Note that if $Au = 0$ and $Av = 0$ then

$$A(cu + dv) = cAu + dAv = c0 + d0 = 0$$

thus it is correctly called a subspace. Also, $u = 0$ is trivially in the nullspace. Example

$$A = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad v_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad v_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

We see that $Av_1 = 0$ and $Av_2 = 0$. What about $v_1 + 3v_2$?

$$A(v_1 + 3v_2) = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Mathematics Review

Column Space: subspace, $\mathcal{R}(A)$, all $\{Ax\}$ (range of A).
Using the last A as the working example:

$$A = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

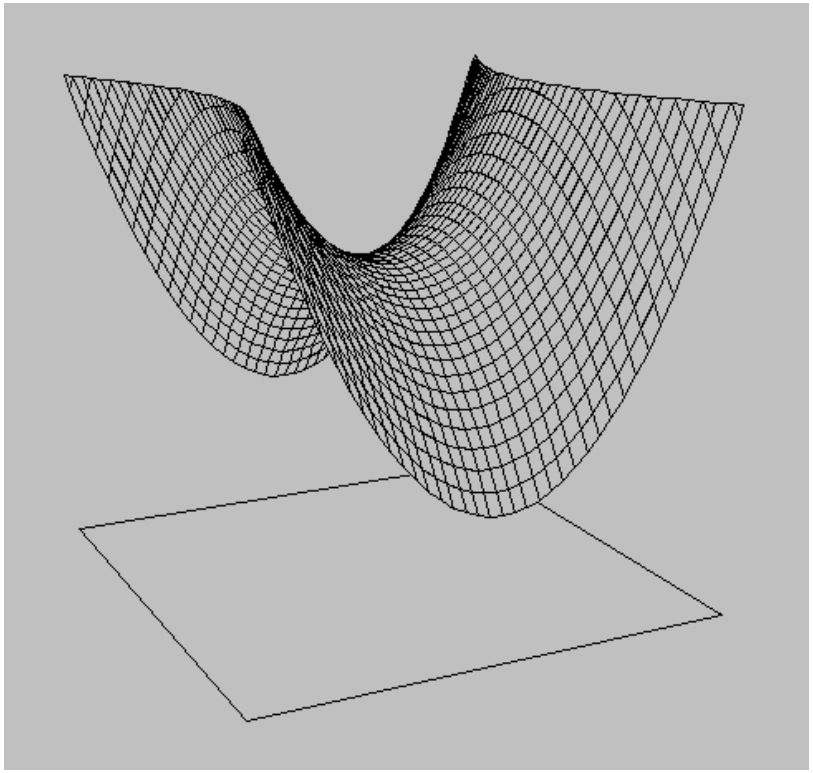
the range is given by the span of the columns. So we have

$$\left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \right\}$$

Note that a similar notion is the span of the rows, called the row space.

Mathematics Review

Manifold: structure, M, which is locally Euclidean (generalized surface).



Metric: measure of distance, $d(x, y)$.

Norm: measure of vector length, $\|v\|$.

Eigenvalues and Eigenvectors

Let x solve $Ax = \lambda x$ (invariant directions).

$$Ax - \lambda x = 0 \quad \Rightarrow \quad (A - \lambda I)x = 0 \quad \Rightarrow \quad x \in \mathcal{N}(A - \lambda I)$$

$$\det(A - \lambda I) = 0 \quad \Rightarrow \quad \lambda$$

Thus we can obtain (λ, x). Known as an eigenvalue, eigenvector pair.

For A with distinct eigenvalues (actually more general than this):

$$A = S \Lambda S^{-1}$$

$$\Lambda = \begin{pmatrix} \lambda_1 & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & \lambda_n \end{pmatrix} \quad S = \begin{pmatrix} x_1 | & \dots & | x_n \end{pmatrix}$$

Orthogonal

Dot product of two vectors x and y: $x \cdot y$

If $x \cdot y = 0$, then we say that x and y are orthogonal.

If a matrix satisfies $Q^{-1} = Q^T$ then we say Q is orthogonal. The columns are mutually orthonormal: orthogonal and unit norm.

If A is symmetric then $A = Q\Lambda Q^T$ where Q is orthogonal.

Do you do this by hand? The original engine is called LINPACK, now LAPACK. C versions of LAPACK exist: GSL - Gnu Scientific Library. You can use GSL to compute all these objects.

Probability

Let X denote a random variable.

Let P denote the probability that X takes on a specific value x: $P(X = x)$.

If X takes on discrete values we say that X is discrete variable.

If X takes on continuous values we say that X is continuous variable.

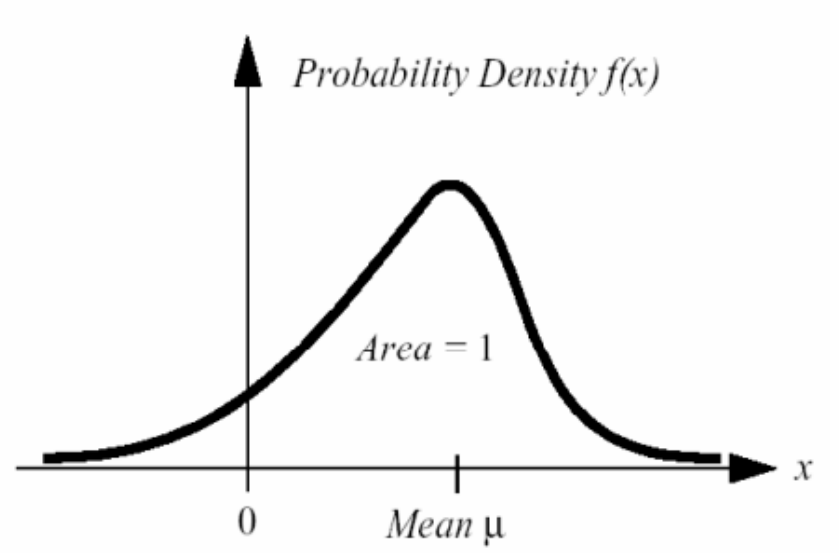
Normally $P(X = x)$ makes sense for discrete spaces and we use $P(x_1 < X < x_2)$ for continuous spaces.

For continuous spaces we define the probability density function (pdf) $p(x)$:

$$P(x_1 \leq X \leq x_2) = \int_{x_1}^{x_2} p(x)\, dx$$

Uncertainty

A probability distribution function:



Distributions

We define the cummulative distribution

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} p(t)\, dt$$

Many distributions exist. One common one is the normal distribution

$$p(x) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-(x-\mu)^2/2\sigma^2}$$

where μ is the mean and σ^2 is the variance (σ is the standard deviation). For multivariate distributions (vector valued random variables) we can exend to

$$p(x) = \frac{1}{(2\pi)^{n/2}\sqrt{\det(\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}$$

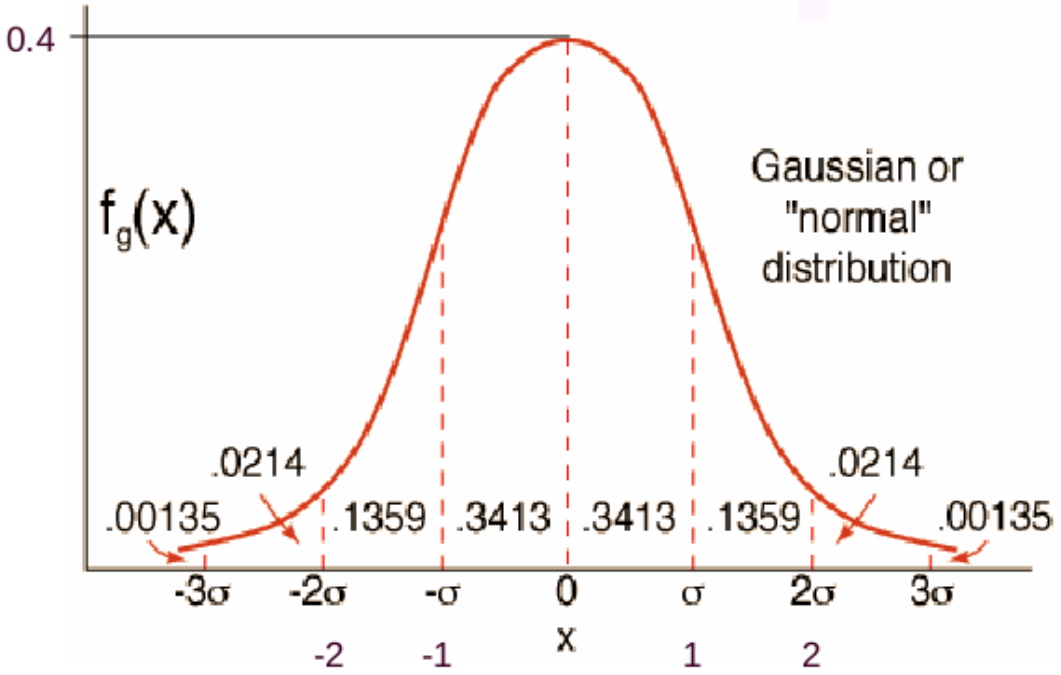
where μ - mean vector, Σ - covariance matrix (symmetric positive definite).

Uncertainty

Normal distribution:

$\mu = 0$ and $\sigma = 1$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



Distributions

Let X, Y be two random variables, the joint distribution is

$$P(x, y) = P(X = x \text{ and } Y = y).$$

We say the the variables are independent if

$$P(x, y) = P(x)P(y)$$

Conditional probability: what is the probability of x if we know y has occurred? Denoted $P(x|y)$,

$$P(x|y) = \frac{P(x, y)}{P(y)}$$

Distributions

If they are independent

$$P(x|y) = \frac{P(x,y)}{P(y)} = \frac{P(x)P(y)}{P(y)} = P(x)$$

Total probability (relax the uppercase formalism)

$$p(x) = \sum_y p(x|y)p(y) \quad \left[= \int_Y p(x|y)p(y)\, dy \right]$$

Bayes Rule (way to invert conditional probabilities)

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Means etc

Expectation (mean or average)

$$E(x) = \sum xp(x) \quad \left[= \int_X xp(x)\, dx \right]$$

Moments

$$\tilde{\mu_r} = E(x^r) = \int_X x^r p(x)\, dx$$

$$\mu = \tilde{\mu_1} = \quad \text{Mean - expected value}$$

Moments about the mean

$$\mu_r = \int_X (x - \mu)^r p(x)\, dx$$

Covariance

Second moment about the mean is called the variance: $\mu_2 = \sigma^2$, where σ is called the standard deviation. Note that variance $= E[(x-\mu)^2]$ and covariance $E(X \cdot Y) - \mu\nu$ where μ, ν are the means for X and Y.

The covariance matrix is given by $\Sigma =$

$$\begin{pmatrix} E[(x_1-\mu_1)(x_1-\mu_1)] & E[(x_1-\mu_1)(x_2-\mu_2)] & \dots & E[(x_1-\mu_1)(x_n-\mu_n)] \\ \dots & \ddots & \dots & \dots \\ E[(x_n-\mu_n)(x_1-\mu_1)] & E[(x_n-\mu_n)(x_2-\mu_2)] & \dots & E[(x_n-\mu_n)(x_n-\mu_n)] \end{pmatrix}$$

$$= E[(x-\mu)(x-\mu)^T]$$

Variance, covariance and crossvariance, variance-covariance...

Covariance

Sample covariance

If you know the population mean, the covariance

$$Q = \frac{1}{N}\sum_{k=1}^{N}(x_k - E(x))(x_k - E(x))^T$$

and if you don't know the mean

$$Q = \frac{1}{N-1}\sum_{k=1}^{N}(x_k - \overline{x})(x_k - \overline{x})^T$$

Note: $(x_1 - \overline{x})$, $(x_2 - \overline{x})$, $(x_2 - \overline{x})$ has $n-1$ residuals (since they sum to zero).

Frames

Represent a rigid body in space.

Define the principle direction (front) as the approach direction - vector a.

A second orthogonal direction to a, the normal to a is n.

A third direction selected $o = a \times n$.

We can load into a matrix

$$F = \begin{pmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{pmatrix}$$

Frames

We can see that this acts like a coordinate system, let $c = [c_1, c_2, c_3]$

$$c' = Fc = c_1 n + c_2 o + c_3 a$$

and F transforms from one coordinate system to another.

F can generate scalings, rotations, reflections, shears.

Does not translate since $F0 = 0$. To translate we need

$$c' = Fc + T$$

Known as an affine map.

Frames

An translation or offset can be described by using a homogenous coordinate system.

$$F = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Simple motion

Homogeneous coordinates

$$\xi = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Allows for general transforms: $\xi' = A\xi$, thus,

$$\xi' = \begin{pmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \xi$$

Homogeneous Coordinates

Rotations - Rotation matrix

- About z

$$R_z = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- About x

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- About y

$$R_y = \begin{pmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Homogeneous Coordinates

Translation - Translation matrix

$$T = \begin{pmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Successive motion can be computed by matrix multiplication.
Let R be a rotation and T be a translation. Then

$$M = TR$$

is the matrix that describes the rotation by R followed by translation by T.

Homogeneous Coordinates: Example

Assume that you are given the following motions: Rotate about the x-axis 30 degrees, translate in y by 3cm, and rotate about the z axis 45 degrees. Find the coordinate transformation.[1]

$$R_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 30 & -\sin 30 & 0 \\ 0 & \sin 30 & \cos 30 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad R_2 = \begin{pmatrix} \cos 45 & -\sin 45 & 0 & 0 \\ \sin 45 & \cos 45 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

[1] the text has examples as well

Homogeneous Coordinates: Example

Then the transformation is $M = R_2 T R_1$

$$= \begin{pmatrix} \cos 45 & -\sin 45 & 0 & 0 \\ \sin 45 & \cos 45 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 30 & -\sin 30 & 0 \\ 0 & \sin 30 & \cos 30 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos 45 & -\sin 45 & 0 & 0 \\ \sin 45 & \cos 45 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 30 & -\sin 30 & 3 \\ 0 & \sin 30 & \cos 30 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} \cos 45 & -\sin 45 \cos 30 & -\sin 45 \sin 30 & -3 \sin 45 \\ \sin 45 & \cos 45 \cos 30 & -\cos 45 \sin 30 & 3 \cos 45 \\ 0 & \sin 30 & \cos 30 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$