Contents

1 About librobotics 2

2 Reference 2

2.1 chi2invtable ................................. 2
2.2 compound .................................... 3
2.3 diffangle .................................... 4
2.4 normangle .................................... 4
2.5 drawarrow .................................... 5
2.6 drawellipse ................................... 6
2.7 drawlabel ..................................... 7
2.8 drawprobellipse .............................. 8
2.9 drawreference ................................ 9
2.10 drawrobot ................................... 10
2.11 drawrect ..................................... 12
2.12 drawtransform ............................... 13
2.13 icompound .................................. 14
2.14 j1comp ....................................... 15
2.15 j2comp ....................................... 15
2.16 jinv ......................................... 16
2.17 mahalanobis .................................. 17
2.18 meanwm ...................................... 18
1 About librobotics

For the sake of brevity, here just the facts:

- librobotics is a small library with frequently used Octave/Matlab functions in Robotics, especially for visualization.
- All commands are fully documented, just type `help command`.
- librobotics is compatible with both, Matlab and Octave.
- It’s open source, feel free to distribute and extend.
- librobotics was written by Kai Arras mainly in 2003-4 as part of the CAS Robot Navigation Toolbox. Minor adaptations since then.
- librobotics can be downloaded from the Social Robotics Lab homepage at [http://srl.informatik.uni-freiburg.de/downloads](http://srl.informatik.uni-freiburg.de/downloads)

Enjoy!

2 Reference

2.1 `chi2invtable`

Lookup table of the inverse of the $\chi^2$ cumulative distribution function.

$x = \text{chi2invtable}(p,v)$ returns the inverse of the $\chi^2$ cumulative distribution function (cdf) with $v$ degrees of freedom at the value $p$. The $\chi^2$ cdf with $v$ degrees of freedom, is the $\Gamma$ cdf with parameters $v/2$ and $2$.

Opposed to `chi2inv` of the Matlab statistics toolbox (which might be not part of your Matlab installation), `chi2invtable` is a lookup table and thereby much faster than `chi2inv`. However, as any lookup table is a collection of sample points, accuracy is smaller. Between the sample points of the cdf, a linear interpolation is made.

Currently, the function supports the degrees of freedom $v$ between 1 and 10 and the probability levels $p$ between 0 and 0.9999 in steps of 0.0001 plus the level 0.99999.

**Example:** A typical usage scenario of `chi2invtable` (or `chi2inv`) is during the matching step of a Kalman filter localization or slam cycle. Given the probability $\alpha$ and features of dimension $n$, `chi2invtable` yields the maximal Mahalanobis distance (or gate distance) $\chi^2_{n,\alpha}$ which a candidate pairing with innovation $\nu_{ij}$ and innovation covariance $S_{ij}$ may have in order to be accepted. In other words, the pairing is accepted if the following holds:

$$\nu_{ij}^T S_{ij} \nu_{ij} < \chi^2_{n,\alpha}$$

See also `chi2inv`.  

2
2.2 compound

Compound relationship in 2D.

\( x_{ik} = \text{compound}(x_{ij}, x_{jk}) \) returns the compound relationship of the two 2D transforms \( x_{ij} \) and \( x_{jk} \) which are arranged head-to-tail. \( x \)'s are 3 \( \times \) 1-vectors \([x, y, \theta]^T\), orientations within \([0, 2\pi]\).

**Example**: Given the transform \( x_{WR} \) which expresses entity \( R \) in the reference frame of \( W \) and transform \( x_{RS} \) which represents entity \( S \) in the frame of \( R \), then the composition \( x_{WS} \) is the relationship which expresses \( S \) in the frame of \( W \):

\[
x_{WS} = x_{WR} \oplus x_{RS}
\]

Note that the compound operation would be the same as vector addition if there were no orientations.

![compound.m](attachment:image.png)

Figure 1: `compound.m`


See also `icomound`, `j1comp`, `j2comp`.
2.3 diffangle

Take difference of two angles and unwrap it.

\[ \alpha = \text{diffangle}(\alpha_1, \alpha_2) \]
determines the minimal difference \(\alpha = \alpha_1 - \alpha_2\) between two angles \(\alpha_1\) and \(\alpha_2\). If either \(\alpha_1\) or \(\alpha_2\) is \(\text{Inf}\), \(\text{Inf}\) is returned.

**Example:** The difference of \(\alpha_1 = 107.54\) and \(\alpha_2 = -115.97\) is \(\alpha = -136.49\).

![Diagram showing angles and difference](image)

Figure 2: diffangle.m

See also normangle.

2.4 normangle

Put angle into a \(2\pi\) interval.

\[ a_r = \text{normangle}(a, \text{min}) \]
puts angle \(a\) into the interval \([\text{min}, \text{min}+2\pi]\). If \(a\) is \(\text{Inf}\), \(\text{Inf}\) is returned.

See also diffangle.
2.5  drawarrow

Draw an arrow.

drawarrow(xs,xe,filled,hsize,color) draws an arrow from xs to xe. The first two elements of xs, xe are interpreted as the x- and y-positions. filled enables and disables head filling, hsize scales the size of the head in [m], and color is a [r g b]-vector or a color string such as 'r' or 'g'.

h = drawarrow(...) return a column vector of handles to the graphic objects of the arrow drawing.

Example: The commands

```matlab
    drawarrow([1 3 -pi/1.8],[2 0 pi/30],0,1,'k');
drawarrow([-3 2 -pi/8],[-2 1 3*pi/2],0,0.3,'b');
drawarrow([0.5 -0.5],[0 2.2],1,1,[0.4 0.9 0.1]);
drawarrow([-1 2], [-2 -1],1,0.2,[0.6 0.6 0.2]);
h = drawarrow([0 -1],[-1 -1.6],1,0.5,'r');
set(h,'LineWidth',3);
```

generate the arrows shown in the figure below. Note that the line width of the last arrow in the bottom of the figure has been changed using the handle vector h.

![Figure 3: drawarrow.m](image)

See also drawreference, plot.
2.6 drawellipse

Draw ellipse.

drawellipse(x,a,b,color) draws an ellipse at $x = [x, y, \theta]^T$ with half axes $a$ and $b$. Orientation $\theta$ is the inclination angle of $a$, regardless if $a$ is smaller or greater than $b$. color is a $[r\ g\ b]$-vector or a color string such as 'r' or 'g'.

$h = \text{drawellipse}(\ldots)$ returns the graphic handle $h$.

**Example:** The commands

```matlab
plot(1,-0.4,'k+');
drawellipse([1 -0.4 pi/6],1,0.5,'k');
plot(0.5,0,'k+');
drawellipse([0.5 0 pi/6],0.25,0.5,'b');
plot(1.8,-0.8,'k+');
h = drawellipse([1.8 -0.8 pi/6],0.2,0.2,[0 0.7 0.3]);
set(h,'LineStyle','-.');
```

generate the ellipses shown below. Note that the line style of the circle can be changed using the handle vector $h$.

![Figure 4: drawellipse.m](image)

See also drawprobellipse.
2.7 drawlabel

Draw scalable text.

drawlabel(x,str, scale, offset, color) draws scalable text str at pose x = [x, y, θ]T imitating the OCR font. With scale = 1, the height of the letters is 1 meter. offset shifts the text in [m] from the x, y position in positive x- and y-direction. color is either a [r g b]-vector or a color string such as ‘r’ or ‘g’.

Currently, the following characters are implemented: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, W, R, S, E, F, M, P.

h = drawlabel(...) returns a column vector of handles to all line objects of the drawing, one handle per line.

Example: The commands

    plot(-3,2,'k+'); drawlabel([-3 2 0],'123',0.5,0.0,'k');
    plot(-3,1,'k+'); drawlabel([-3 1 0],'123',0.5,0.1,'k');
    plot(-3,0,'k+'); drawlabel([-3 0 0],'123',0.5,0.2,'k');
    plot(-1,2,'k+'); drawlabel([-1,2 -2.8],'R1',0.2,0.1,[.5 .5 .5]);
    plot( 1,3,'k+'); drawlabel([ 1 3 -pi/1.8],'F30493',0.8,0.3,'g');

    plot(-2,-1,'k+'); h = drawlabel([-2 -1 2.6],'S04',0.3,0.1,'r');
    set(h,'LineWidth',3);
    plot(-1,-2,'k+'); h = drawlabel([-1 -2 pi/3],'REF',0.3,0.1,'m');
    set(h,'LineWidth',2);

generate the figure shown below. Note that line width can be varied using the handle vector h.

![Figure 5: drawlabel.m](image-url)

See also text.
2.8 drawprobellipse

Draw elliptic probability region of a Gaussian in 2D.

drawprobellipse(x,C,alpha,color) draws the elliptic iso-probability contour of a Gaussian distributed bivariate random vector \( \mathbf{x} \) at the significance level \( \alpha \). The ellipse is centered at \( \mathbf{x} = [x, y]^T \) where \( \mathbf{C} \) is the associated \( 2 \times 2 \) covariance matrix. \( \text{color} \) is a \([r \ g \ b]\)-vector or a color string such as ‘r’ or ‘g’.

\( \mathbf{x} \) and \( \mathbf{C} \) can also be of size \( 3 \times 1 \) and \( 3 \times 3 \) respectively.

The function uses \( \text{chi2invtable} \) instead of \( \text{chi2inv} \) from the Matlab statistics toolbox.

In case of a negative definite matrix \( \mathbf{C} \), the ellipse collapses to a line which is drawn instead.

\( h = \text{drawprobellipse}(...) \) returns the graphic handle \( h \).

**Example:** The commands

\[
\begin{align*}
    x1 & = [1, 2]; C1 = [0.25 -0.2; -0.2 0.3]; \\
    \text{drawprobellipse}(x1,C1,0.95,'k'); \\
    x2 & = [-1,0]; C2 = [0.3 -0.03; -0.03 0.01]; \\
    \text{drawprobellipse}(x2,C2,0.95,'k'); \\
    x3 & = [-2,3]; C3 = [0.01 0.005; 0.005 0.015]; \\
    \text{drawprobellipse}(x3,C3,0.95,'k');
\end{align*}
\]

generate the ellipses shown below. Note that line width can be varied using the handle vector \( h \).

![Figure 6: drawprobellipse.m](Image)

See also \texttt{drawellipse}, \texttt{chi2invtable}, \texttt{chi2inv}.
2.9 drawreference

Draw coordinate reference frame.

drawreference(x,label,size,color) draws a reference frame at pose \( x = [x, y, \theta]^T \) and labels it with the string label. size is the length of the frame axes in [m], and color is a [r g b]-vector or a color string such as ‘r’ or ‘g’. 

\[ h = \text{drawreference}(...) \] returns a column vector of handles to all graphic objects of the drawing. Remember that not all graphic properties apply to all types of graphic objects. Use \text{findobj} to find and access the individual objects.

Example: The commands

\[
\begin{align*}
\text{drawreference}([0 2 \pi/8], 'W', 1, 'k'); \\
\text{drawreference}([0 0 \pi/3], 'R2', 0.4, 'b'); \\
\text{drawreference}([0.8 1.3 -1.8], 'S43', 0.5, [.8 .5 .1]); \\
\text{drawreference}([2 2.3 -0.3], '', 0.6, [.6 .6 .6]); \\
\end{align*}
\]

\[ h = \text{drawreference}([2 0.7 \pi/9], '98', 0.8, [.3 .7 .3]); \]
\[
\text{set}(h, '\text{LineWidth}', 2);
\]

generate the figure shown below. Note that line width, line style and color can be varied using the handle vector \( h \).

![Figure 7: drawreference.m](image)

See also drawarrow, drawlabel, findobj, plot.
2.10 *drawrobot*

Draw robot.

drawrobot(*x*,*color*) draws a robot at pose \( x = [x, y, \theta]^T \) such that the robot reference frame is attached to the center of the wheelbase with the \( x \)-axis looking forward. *color* is a \([r g b]\)-vector or a color string such as ‘r’ or ‘g’.

drawrobot(*x*,*color*,*type*) draws a robot of type *type*. Five different models are implemented:

- **type** = 0 draws only a cross with orientation \( \theta \)
- **type** = 1 is a differential drive robot without contour
- **type** = 2 is a differential drive robot with round shape
- **type** = 3 is a round shaped robot with a line at \( \theta \)
- **type** = 4 is a differential drive robot with rectangular shape
- **type** = 5 is a rectangular shaped robot with a line at \( \theta \)

drawrobot(*x*,*color*,*type*,*w*,*l*) draws a robot of type *type* with width *w* and length *l* in [m].

\( h = \text{drawrobot}(...) \) returns a column vector of handles to all graphic objects of the robot drawing. Remember that not all graphic properties apply to all types of graphic objects. Use *findobj* to find and access the individual objects.

---

**Figure 8: drawrobot.m**

---

**Example:** The commands

```plaintext
drawrobot([0 1 1.5],’k’,0);
drawrobot([1 1 1.4],’k’,1);
drawrobot([2 1 1.3],’k’,2);
drawrobot([3 1 1.2],’k’,3);
drawrobot([4 1 1.1],’k’,4);
drawrobot([5 1 1.0],’k’,5);
drawrobot([0 -0.8 2.0],’r’,2,0.6,0.6);
drawrobot([1 -1.2 1.9],[.5 .5 .4],4,0.5,0.3);
drawrobot([2 -0.8 1.8],[.7 .5 .4],1,0.2,0.8);
```

10
drawrobot([3 -1.2 1.7],[.9 .5 .4],5,0.3,0.7);

h = drawrobot([4 -0.8 1.6],'g',3,0.4,0.1);
set(h,'LineWidth',3);

h = drawrobot([5 -1.2 1.5],'k',0,0.4,0.1);
set(h,'LineWidth',2,'LineStyle',':');

generate the robots shown above. Note that line width, line style and color can be varied using the handle vector h.

See also drawrect, drawarrow, findobj, plot.
2.11 drawrect

Draw rounded rectangle.

drawrect(x,w,h,r,filled,color) draws a rectangle with round corners of radius r, width w and height h, centered at pose x where x is the 3 × 1 vector [x,y,θ]T. With filled = 1 the rectangle is filled with color color, with filled = 0 only the contour is drawn. color is a [r g b]-vector or a Matlab color string such as 'r' or 'g'.

Note that 2r must be greater or equal than the smaller of the two values w, h. For 2r = w = h, drawrect draws a circle.

h = drawrect(...) returns the graphic handle h.

Example: The commands

drawrect([0.4 1 2.6],1,0.6,0.2,0,'b');
drawrect([1.9 2 2.5],1,1.2,0.1,1,[.4 .9 .0]);
drawrect([1.9 2 2.5],1,1.2,0.1,0,([2 .7 .0]));
drawrect([3.0 1 2.4],0.2,1.7,0,0,0,'r');
drawrect([4.0 0 2.3],0.4,0.4,0.2,1,[.8 .8 .8]);

h = drawrect([4 0 2.3],0.4,0.4,0.2,0,'k');
set(h,'LineWidth',2);
h = drawrect([5 1 0.6],0.3,0.7,0.15,0,[.9 .7 .0]);
set(h,'LineWidth',4);

generates the figure shown below. Note that line width, line style and color can be varied using the handle vector h.

Figure 9: drawrect.m

See also drawreference, plot.
2.12 drawtransform

Illustrates a spatial relationship.

drawtransform(xs,xe,shape,label,color) draws a nice looking curved arrow from location $xs$ ($3 \times 1$) to $xe$ ($3 \times 1$) and labels it with the string $label$. $color$ is a [r g b]-vector or a Matlab color string such as 'r' or 'g'. $shape$ controls the shape of the curve: '/' for a S-shape, '\ for a Z-shape, '(' for a left arc and ')' for a right arc.

$h = \text{drawtransform}(\ldots)$ returns a column vector of handles to all graphic objects of the drawing. Remember that not all graphic properties apply to all types of graphic objects.

Example: The commands

```matlab
plot(0,0,'k+'); plot(0.2, 1.5,'k+');
drawtransform([0 0],[0.2 1.5],'/','x1',[.9 .8 .0]);
plot(1,0,'k+'); plot(1.2, 1.5,'k+');
drawtransform([1 0],[1.2 1.5],'/','x2',[.7 .6 .0]);
plot(2,0,'k+'); plot(2.2, 1.5,'k+');
drawtransform([2 0],[2.2 1.5],'/','x3',[.5 .3 .0]);
plot(3,0,'k+'); plot(3.2, 1.5,'k+');
drawtransform([3 0],[3.2 1.5],')','x4',[.3 .0 .0]);
```

$h = \text{drawtransform}(\ldots)$, set(h,'LineStyle','--');

$h = \text{drawtransform}(\ldots)$, set(h,'LineStyle',':');

generate the arrows shown below. Note that line width, line style and color can be set using the handle vector $h$.

![Figure 10: drawtransform.m](image)

See also drawreference, plot.
2.13 icompound

Inverse 2D relationship.

\[ x_{ji} = \text{icompound}(x_{ij}) \] returns the inverted 2D transform \( x_{ji} \) given the relationship \( x_{ij} \). All \( x \)'s are 3 \( \times \) 1-vectors, all angles within \([0..2\pi] \).

**Example:** Given a feature with attached frame \( F \) and the robot with attached frame \( R \) represented both in the world reference frame \( W \) by the transforms \( x_{WF} \) and \( x_{WR} \), we look for \( F \) expressed in the robot reference frame \( R \). The solution is the composition of the inverted \( x_{WR} \) with \( x_{WF} \):

\[ x_{RF} = \ominus x_{WR} \oplus x_{WF} \]  

(1)

This frame transform is usually called *measurement prediction* in the context of feature-based Kalman filter localization or slam (with the simplifying assumption that sensor frame and robot frame coincide).

The figure has been generated by the following code:

```matlab
xwr = [1.0, 2, -1.4];
xwf = [3.2, 1, pi/9];
drawreference([0 0 -pi/19], 'W', 0.7, 'k');
drawreference(xwr, 'R', 0.7, [0.7 0 0]);
drawrobot(xwr, 'k');
drastransform(zeros(3,1), xwr, '\', 'x_W_R', 'k');
drawreference(xwf, 'F', 0.7, [0 0.7 0]);
drastransform(zeros(3,1), xwf, '/', 'x_W_F', 'k');
drastransform(xwr, xwf, '/', 'x_R_F', 'k');
```

**Figure 11: icompound.m**


See also compound, jinv.
2.14 j1comp

First Jacobian of the compound operator.

\[ J = j1comp(\mathbf{x}_i, \mathbf{x}_j) \]
returns the Jacobian matrix of the 2D composition of \( \mathbf{x}_i \)
and \( \mathbf{x}_j \) derived with respect to the first operand \( \mathbf{x}_i \). All \( \mathbf{x} \)'s are 3 \times 1-vectors, \( J \)
is a 3 \times 3-matrix.

The Jacobian is used to perform first-order error propagation when the input transforms \( \hat{\mathbf{x}}_{ij} \) and \( \hat{\mathbf{x}}_{jk} \) of the composition \( \hat{\mathbf{x}}_{ik} = \hat{\mathbf{x}}_{ij} \oplus \hat{\mathbf{x}}_{jk} \) are uncertain. To
calculate the uncertainty of the output \( \hat{\mathbf{x}}_{ik} \), the compound operator is derivated
with respect to the two operands yielding a 3 \times 6 Jacobian matrix \( J_\oplus \)
which consists of a left 3 \times 3 half, \( J_{1\oplus} \), and a right 3 \times 3 half, \( J_{2\oplus} \).

\[
J_{1\oplus} = \frac{\delta \hat{\mathbf{x}}_{ik}}{\delta \mathbf{x}_{ij}} \mid \hat{\mathbf{x}}_{ij} \quad J_{2\oplus} = \frac{\delta \hat{\mathbf{x}}_{ik}}{\delta \mathbf{x}_{jk}} \mid \hat{\mathbf{x}}_{jk}
\]

\[
J_\oplus = [J_{1\oplus} \quad J_{2\oplus}]
\]

With \( C_{ijk} \) as the input covariance matrix

\[
C_{ijk} = \begin{bmatrix} C_{ij} & C_{ijjk} \\ C_{jki} & C_{jk} \end{bmatrix}
\]

the covariance matrix of the output transform \( C_{ik} \) is given by the error propagation law

\[
C_{ik} = J_\oplus C_{ijk} J_\oplus^T = J_{1\oplus} C_{ij} J_{1\oplus}^T + J_{1\oplus} C_{ijjk} J_{2\oplus}^T + J_{2\oplus} C_{jki} J_{1\oplus}^T + J_{2\oplus} C_{jk} J_{2\oplus}^T
\]

where the submatrix \( C_{ijjk} \) (\( = C_{jki}^T \)) is the cross-correlation between \( \hat{\mathbf{x}}_{ij} \) and \( \hat{\mathbf{x}}_{jk} \).

See also \( j2comp, jinv, compound, icompound \).

2.15 j2comp

Second Jacobian of the compound operator.

\[ J = j2comp(\mathbf{x}_i, \mathbf{x}_j) \]
returns the Jacobian matrix of the 2D composition of \( \mathbf{x}_i \)
and \( \mathbf{x}_j \) derived with respect to the second operand \( \mathbf{x}_j \). All \( \mathbf{x} \)'s are 3 \times 1-vectors, \( J \)
is a 3 \times 3-matrix.

See explanation at \( j1comp \).

See also \( j1comp, jinv, compound, icompound \).
2.16 jinv

Jacobian of the inverse compound operator.

\( J = jinv(x_{ij}) \) returns the Jacobian matrix of the inversion \( x_{ji} \) of \( x_{ij} \). All \( x \)'s are \( 3 \times 1 \)-vectors.

The Jacobian is used to perform first-order error propagation when the input transform \( \hat{x}_{ij} \) of the inversion \( \hat{x}_{ji} = \ominus \hat{x}_{ij} \) is uncertain. To calculate the uncertainty of the output \( \hat{x}_{ji} \), the inverse compound operator is derivated with respect to the operand yielding the \( 3 \times 3 \) Jacobian matrix \( J_\ominus \).

\[
J_\ominus = \left. \frac{\delta \hat{x}_{ji}}{\delta x_{ij}} \right|_{\hat{x}_{ij}}
\]

With \( C_{ij} \) as the input covariance matrix, the covariance matrix of the output transform, \( C_{ji} \), is given by the error propagation law

\[
C_{ji} = J_\ominus C_{ij} J_\ominus^T
\]

See also \texttt{j1comp, j2comp, compound, icompound}. 
2.17 mahalanobis

Calculate the Mahalanobis distance.

\( d = \text{mahalanobis}(v, S) \) calculates the chi square distributed Mahalanobis distance given the innovation vector \( v \) and the innovation covariance matrix \( S \).

Formally, with the innovation \( \nu \) and the innovation covariance matrix \( S \), the Mahalanobis distance is

\[
D = \nu^T S \nu
\]  

(2)

The Mahalanobis distance is a quadratic form and allows to test on positive definiteness of a matrix \( S \). With any \( \nu \neq 0 \), \( S \) is positive definite if \( D > 0 \) and positive semidefinite if \( D \geq 0 \).

See also \texttt{chi2invtable}, \texttt{chi2inv}.
2.18 meanwm

Multivariate weighted mean (Information Filter).

\[ [x_w, C_w] = \text{meanwm}(x, C) \]

calculates the multivariate weighted mean. \( x \) is a matrix of dimension \( m \times n \) where each column is interpreted as a Gaussian distributed random vector of dimension \( m \times 1 \). \( C \) is a \( m \times m \times n \) matrix where each \( m \times m \) matrix is interpreted as the covariance estimate associated to its respective row vector. The function returns the weighted mean vector \( x_w \) and the weighted covariance matrix \( C_w \) of dimensions \( m \times 1 \) and \( m \times m \) respectively.

The multivariate weighted mean is also known as the Information Filter (IF), a batch formulation of the (recursive) Kalman filter. Given the random vectors \( x_1, x_2, \ldots, x_n \) with associated covariance matrices \( C_1, C_2, \ldots, C_n \), the IF calculates

\[
\begin{align*}
x_w &= C_w \sum C_i^{-1} x_i \\
C_w^{-1} &= \sum C_i^{-1}
\end{align*}
\]

Example: The Matlab code

```matlab
x1 = [1.5; 1]; C1 = [0.08 -0.082; -0.082 0.1];
drawprobellipse(x1,C1,0.95,[.9 .5 0]);
plot(x1(1),x1(2),'+','Color',[.9 .5 0]);
x2 = [1; 0]; C2 = [0.18 -0.02; -0.02 0.01];
drawprobellipse(x2,C2,0.95,[.9 .2 0]);
plot(x2(1),x2(2),'+','Color',[.9 .2 0]);
xin = cat(2,x1,x2); Cin = cat(3,C1,C2);
[xw,Cw] = meanwm(xin,Cin);
drawprobellipse(xw,Cw,0.95,'k');
plot(xw(1),xw(2),'k+');
x3 = [3; 3]; C3 = [0.01 0.005; 0.005 0.015];
drawprobellipse(x3,C3,0.95,[0 .9 .7]);
plot(x3(1),x3(2),'+','Color',[0 .9 .7]);
```

Figure 12: meanwm.m
generates the figure shown below. Note how the command `cat` is used to prepare the input arguments.

See also `mean`, `cat`, `drawprobellipse`. 