ベクトル群の重ね合わせ

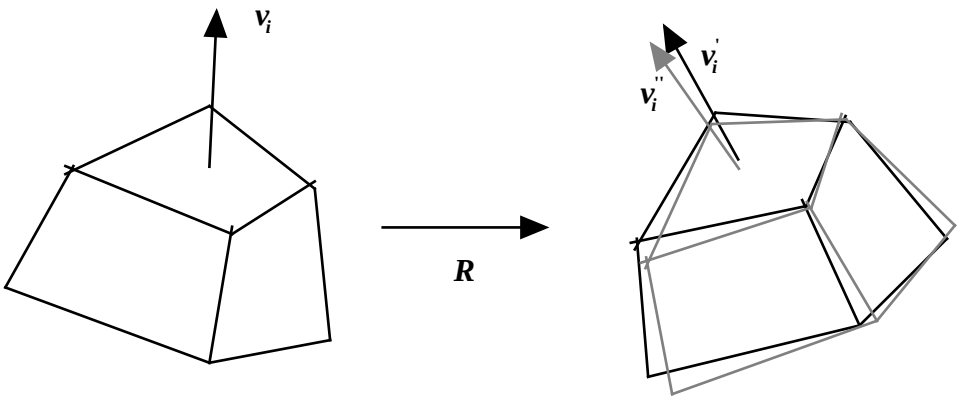
大島正毅

初版　　2001.02.19
最終改訂日　2001.03.21

ビジョン処理において、立体と立体との重ね合わせは重要なテーマである。ある既知の物体が存在するとき、観測されたデータの中にその物体があることは分かっているが、その位置・姿勢は不明であるという状況が考えられる。これをいかに解決するかはビジョン処理の一般的問題（例　[1]）であるが、ここではその部分問題として姿勢だけを問題にする。すなわち、あるベクトル群と別のベクトル群を重ね合わせる座標変換を求めることを考える。

2つ以上のベクトルよりなるベクトル群が、（各ベクトルのペアとして）対応関係が分かっているベクトル群に一致するような変換を求める。観測データとモデルデータを一致させるような状況を想定すると、観測値に誤差を伴うから、一意には定まらない。また、2つ以上のベクトルよりなるベクトル群を考えるときは過剰拘束である。ここではこのような状況での最適解を求める。

あるベクトル群（$\boldsymbol{v}_i$で代表する）と各ベクトル同士の対応関係の分かっている（回転して）観測されたベクトル群（$\boldsymbol{v}_i'$で代表する）が与えられているものとする（すなわち $\left(\boldsymbol{v}_i\ \boldsymbol{v}_i'\right)\left\{1,2,\quad,N\right\}$）。回転$\boldsymbol{R}$が未知としてこれを求めたい。ベクトルとしては単位ベクトルを考える。



上図で$\boldsymbol{v}_i$はモデルパターン、$\boldsymbol{v}_i'$は観測パターン、$\boldsymbol{v}_i'' = \boldsymbol{R}\,\boldsymbol{v}_i$は$\boldsymbol{R}$によって$\boldsymbol{v}_i$から得られたパターンとする。これがなるべく$\boldsymbol{v}_i'$と一致するように$R$を定めるものとする。

$\boldsymbol{v}_i'' - \boldsymbol{v}_i' = R\boldsymbol{v}_i - \boldsymbol{v}_i'$は回転後の$\boldsymbol{v}_i$と$\boldsymbol{v}_i'$の差を意味するから、その絶対値によって評価できる。よって、

$$E = \sum_{i=1}^{N} \left| R\boldsymbol{v}_i - \boldsymbol{v}_i' \right|^2 \qquad (1)$$

を評価尺度として最適な$\boldsymbol{R}$を求めるものとする。

$$R=\begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \qquad (2)$$

として、

$$R\boldsymbol{v}-\boldsymbol{v}'=\begin{pmatrix} R_{11}v_x+R_{12}v_y+R_{13}v_z-v'_x \\ R_{21}v_x+R_{22}v_y+R_{23}v_z-v'_y \\ R_{31}v_x+R_{32}v_y+R_{33}v_z-v'_z \end{pmatrix} \qquad (3)$$

（ここに$\boldsymbol{v}_i$、$\boldsymbol{v}_i'$のiは省略する。以下同様）

よって

$$E=\sum_i \left\{ \left(R_{11}v_x+R_{12}v_y+R_{13}v_z-v'_x\right)^2+\left(R_{21}v_x+R_{22}v_y+R_{23}v_z-v'_y\right)^2+\left(R_{31}v_x+R_{32}v_y+R_{33}v_z-v'_z\right)^2 \right\} \qquad (4)$$

後で用いるため、整理すると

$$\begin{aligned} E=&\left(R_{11}^2v_x^2+R_{12}^2v_y^2+R_{13}^2v_z^2+v_x'^2+2R_{11}R_{12}v_xv_y+2R_{11}R_{13}v_xv_z+2R_{12}R_{13}v_yv_z-2R_{11}v_xv'_x-2R_{12}v_yv'_x-2R_{13}v_zv'_x\right) \\ &+\left(R_{21}^2v_x^2+R_{22}^2v_y^2+R_{23}^2v_z^2+v_y'^2+2R_{21}R_{22}v_xv_y+2R_{21}R_{23}v_xv_z+2R_{22}R_{23}v_yv_z-2R_{21}v_xv'_y-2R_{22}v_yv'_y-2R_{23}v_zv'_y\right) \\ &+\left(R_{31}^2v_x^2+R_{32}^2v_y^2+R_{33}^2v_z^2+v_z'^2+2R_{31}R_{32}v_xv_y+2R_{31}R_{33}v_xv_z+2R_{32}R_{33}v_yv_z-2R_{31}v_xv'_z-2R_{32}v_yv'_z-2R_{33}v_zv'_z\right) \end{aligned} \qquad (4')$$

(ここにΣは省略している)

最適のRの近傍に$R'=R+\varepsilon_R$なるRの推測値R'と誤差ε_Rを考える。このとき(4)の{ }内は

$$\begin{aligned} &\left\{\left(R'_{11}-\varepsilon_{R11}\right)v_x+\left(R'_{12}-\varepsilon_{R12}\right)v_y+\left(R'_{13}-\varepsilon_{R13}\right)v_z-v'_x\right\}^2 \\ &+\left\{\left(R'_{21}-\varepsilon_{R21}\right)v_x+\left(R'_{22}-\varepsilon_{R22}\right)v_y+\left(R'_{23}-\varepsilon_{R23}\right)v_z-v'_y\right\}^2 \\ &+\left\{\left(R'_{31}-\varepsilon_{R31}\right)v_x+\left(R'_{32}-\varepsilon_{R32}\right)v_y+\left(R'_{33}-\varepsilon_{R33}\right)v_z-v'_z\right\}^2 \end{aligned} \qquad (5)$$

整理して

$$\left\{\left(\varepsilon_{R11}-R'_{11}\right)v_x+\left(\varepsilon_{R12}-R'_{12}\right)v_y+\left(\varepsilon_{R13}-R'_{13}\right)v_z+v'_x\right\}^2$$
$$+\left\{\left(\varepsilon_{R21}-R'_{21}\right)v_x+\left(\varepsilon_{R22}-R'_{22}\right)v_y+\left(\varepsilon_{R23}-R'_{23}\right)v_z+v'_y\right\}^2$$
$$+\left\{\left(\varepsilon_{R31}-R'_{31}\right)v_x+\left(\varepsilon_{R32}-R'_{32}\right)v_y+\left(\varepsilon_{R33}-R'_{33}\right)v_z+v'_z\right\}^2$$

$\frac{\partial E}{\partial \varepsilon_R}$は

ε_{R11}について $2v_x\left\{\left(\varepsilon_{R11}-R'_{11}\right)v_x+\left(\varepsilon_{R12}-R'_{12}\right)v_y+\left(\varepsilon_{R13}-R'_{13}\right)v_z+v'_x\right\}$

ε_{R12}について $2v_y\left\{\left(\varepsilon_{R11}-R'_{11}\right)v_x+\left(\varepsilon_{R12}-R'_{12}\right)v_y+\left(\varepsilon_{R13}-R'_{13}\right)v_z+v'_x\right\}$

ε_{R13}について $2v_z\left\{\left(\varepsilon_{R11}-R'_{11}\right)v_x+\left(\varepsilon_{R12}-R'_{12}\right)v_y+\left(\varepsilon_{R13}-R'_{13}\right)v_z+v'_x\right\}$

ε_{R21}について $2v_x\left\{\left(\varepsilon_{R21}-R'_{21}\right)v_x+\left(\varepsilon_{R22}-R'_{22}\right)v_y+\left(\varepsilon_{R23}-R'_{23}\right)v_z+v'_y\right\}$

ε_{R22}について $2v_y\left\{\left(\varepsilon_{R21}-R'_{21}\right)v_x+\left(\varepsilon_{R22}-R'_{22}\right)v_y+\left(\varepsilon_{R23}-R'_{23}\right)v_z+v'_y\right\}$

ε_{R23}について $2v_z\left\{\left(\varepsilon_{R21}-R'_{21}\right)v_x+\left(\varepsilon_{R22}-R'_{22}\right)v_y+\left(\varepsilon_{R23}-R'_{23}\right)v_z+v'_y\right\}$ (6)

ε_{R31}について $2v_x\left\{\left(\varepsilon_{R31}-R'_{31}\right)v_x+\left(\varepsilon_{R32}-R'_{32}\right)v_y+\left(\varepsilon_{R33}-R'_{33}\right)v_z+v'_z\right\}$

ε_{R32}について $2v_y\left\{\left(\varepsilon_{R31}-R'_{31}\right)v_x+\left(\varepsilon_{R32}-R'_{32}\right)v_y+\left(\varepsilon_{R33}-R'_{33}\right)v_z+v'_z\right\}$

ε_{R33}について $2v_z\left\{\left(\varepsilon_{R31}-R'_{31}\right)v_x+\left(\varepsilon_{R32}-R'_{32}\right)v_y+\left(\varepsilon_{R33}-R'_{33}\right)v_z+v'_z\right\}$

（ただしΣを省略している）

これを整理すると：

ε_{R11}について $2\left\{\left(\varepsilon_{R11}-R'_{11}\right)v_x^2+\left(\varepsilon_{R12}-R'_{12}\right)v_xv_y+\left(\varepsilon_{R13}-R'_{13}\right)v_xv_z+v_xv'_x\right\}$

ε_{R12}について $2\left\{\left(\varepsilon_{R11}-R'_{11}\right)v_yv_x+\left(\varepsilon_{R12}-R'_{12}\right)v_y^2+\left(\varepsilon_{R13}-R'_{13}\right)v_yv_z+v_yv'_x\right\}$

ε_{R13}について $2\left\{\left(\varepsilon_{R11}-R'_{11}\right)v_zv_x+\left(\varepsilon_{R12}-R'_{12}\right)v_zv_y+\left(\varepsilon_{R13}-R'_{13}\right)v_z^2+v_zv'_x\right\}$

ε_{R21}について $2\left\{\left(\varepsilon_{R21}-R'_{21}\right)v_x^2+\left(\varepsilon_{R22}-R'_{22}\right)v_xv_y+\left(R'_{23}-\varepsilon_{R23}\right)v_xv_z+v_xv'_y\right\}$

ε_{R22}について　$2\left\{\left(\varepsilon_{R21}-R'_{21}\right)v_y v_x+\left(\varepsilon_{R22}-R'_{22}\right)v_y^2+\left(\varepsilon_{R23}-R'_{23}\right)v_y v_z+v_y v'_y\right\}$　　(7)

ε_{R23}について　$2\left\{\left(\varepsilon_{R21}-R'_{21}\right)v_z v_x+\left(\varepsilon_{R22}-R'_{22}\right)v_z v_y+\left(\varepsilon_{R23}-R'_{23}\right)v_z^2+v_z v'_y\right\}$

ε_{R31}について　$2\left\{\left(\varepsilon_{R31}-R'_{31}\right)v_x^2+\left(\varepsilon_{R32}-R'_{32}\right)v_x v_y+\left(\varepsilon_{R33}-R'_{33}\right)v_x v_z+v_x v'_z\right\}$

ε_{R32}について　$2\left\{\left(\varepsilon_{R31}-R'_{31}\right)v_y v_x+\left(\varepsilon_{R32}-R'_{32}\right)v_y^2+\left(\varepsilon_{R33}-R'_{33}\right)v_y v_z+v_y v'_z\right\}$

ε_{R33}について　$2\left\{\left(\varepsilon_{R31}-R'_{31}\right)v_z v_x+\left(\varepsilon_{R32}-R'_{32}\right)v_z v_y+\left(\varepsilon_{R33}-R'_{33}\right)v_z^2+v_z v'_z\right\}$

Rは直交マトリクスだから、2つのベクトルA、Bの回転前後の角度が不変。

$(RA)^t(RB)=A^tB$　より

$A^tR^tRB=A^tB$　よって

$R^tR=I$　　（Iは単位マトリクス）　これより

$$\begin{aligned}
R_{11}^2+R_{21}^2+R_{31}^2-1&=0\\
R_{11}R_{12}+R_{21}R_{22}+R_{31}R_{32}&=0\\
R_{11}R_{13}+R_{21}R_{23}+R_{31}R_{33}&=0\\
R_{12}^2+R_{22}^2+R_{32}^2-1&=0\\
R_{12}R_{13}+R_{22}R_{23}+R_{32}R_{33}&=0\\
R_{13}^2+R_{23}^2+R_{33}^2-1&=0
\end{aligned}\qquad (8)$$

これらの左辺をϕ_1～ϕ_6とおく。Rの近傍においてϕ_1～ϕ_6を線形近似し右辺=0 とすると：

（ただし、R'_{11}～R'_{33}はRの現在推定値、ϕ'_1～ϕ'_6（R'_{11}～R'_{33}をϕ_1～ϕ_6に代入して求める）はϕ_1～ϕ_6の現在推定値）

$$\begin{aligned}
\phi_1&=\phi'_1-2R'_{11}\varepsilon_{R11}-2R'_{21}\varepsilon_{R21}-2R'_{31}\varepsilon_{R31}=0\\
\phi_2&=\phi'_2-R'_{12}\varepsilon_{R11}-R'_{11}\varepsilon_{R12}-R'_{22}\varepsilon_{R21}-R'_{21}\varepsilon_{R22}-R'_{32}\varepsilon_{R31}-R'_{31}\varepsilon_{R32}=0\\
\phi_3&=\phi'_3-R'_{13}\varepsilon_{R11}-R'_{11}\varepsilon_{R13}-R'_{23}\varepsilon_{R21}-R'_{21}\varepsilon_{R23}-R'_{33}\varepsilon_{R31}-R'_{31}\varepsilon_{R33}=0\\
\phi_4&=\phi'_4-2R'_{12}\varepsilon_{R12}-2R'_{22}\varepsilon_{R22}-2R'_{32}\varepsilon_{R32}=0\\
\phi_5&=\phi'_5-R'_{13}\varepsilon_{R12}-R'_{12}\varepsilon_{R13}-R'_{23}\varepsilon_{R22}-R'_{22}\varepsilon_{R23}-R'_{33}\varepsilon_{R32}-R'_{32}\varepsilon_{R33}=0\\
\phi_6&=\phi'_6-2R'_{13}\varepsilon_{R13}-2R'_{23}\varepsilon_{R23}-2R'_{33}\varepsilon_{R33}=0
\end{aligned}\qquad (9)$$

（9）の拘束条件のもとにEを最小化したい。Lagrangeの未定係数法により

$G=\frac{1}{2}E-\sum_{j=1}^{6}\mu_j\phi_j$　として

ε_Rに関してGを最小化すればよい。（ここにEの前の$\frac{1}{2}$および、Σの前の-は計算を簡素化するための便宜上のもの。必要があればμ_jを読み替えればよいから一般性を失わない）

$\frac{\partial G}{\varepsilon_R}=0$：　$\frac{1}{2}\frac{\partial E}{\partial\varepsilon_R}-\sum_{j=1}^{6}\mu_j\frac{\partial}{\partial\varepsilon_R}\phi_j=0$　　（10）

これは未知数ε_Rについて9個の式である。また未定係数μ_j（6個）も未知である。一方、Rの拘束条件からε_Rについて6つの式（9）が成り立つ。これらを（線形）連立させてε_Rを求めることができる。

解くべき線形連立方程式は（10）と（9）からなる。

（10）の前半は（7）で求まっている。（10）の後半のΣは（9）より

$$
\begin{aligned}
&\varepsilon_{R11}\text{について} \quad -2\mu_1 R'_{11} - \mu_2 R'_{12} - \mu_3 R'_{13} \\
&\varepsilon_{R12}\text{について} \quad -\mu_2 R'_{11} - 2\mu_4 R'_{12} - \mu_5 R'_{13} \\
&\varepsilon_{R13}\text{について} \quad -\mu_3 R'_{11} - \mu_5 R'_{12} - 2\mu_6 R'_{13} \\
&\varepsilon_{R21}\text{について} \quad -2\mu_1 R'_{21} - \mu_2 R'_{22} - \mu_3 R'_{23} \\
&\varepsilon_{R22}\text{について} \quad -\mu_2 R'_{21} - 2\mu_4 R'_{22} - \mu_5 R'_{23} \\
&\varepsilon_{R23}\text{について} \quad -\mu_3 R'_{21} - \mu_5 R'_{22} - 2\mu_6 R'_{23} \\
&\varepsilon_{R31}\text{について} \quad -2\mu_1 R'_{31} - \mu_2 R'_{32} - \mu_3 R'_{33} \\
&\varepsilon_{R32}\text{について} \quad -\mu_2 R'_{31} - 2\mu_4 R'_{32} - \mu_5 R'_{33} \\
&\varepsilon_{R33}\text{について} \quad -\mu_3 R'_{31} - \mu_5 R'_{32} - 2\mu_6 R'_{33}
\end{aligned}
\qquad \text{（11）}
$$

これらから（10）は

$$
\begin{aligned}
\varepsilon_{R11}:&\ \left(\varepsilon_{R11}-R'_{11}\right)v_x^2+\left(\varepsilon_{R12}-R'_{12}\right)v_x v_y+\left(\varepsilon_{R13}-R'_{13}\right)v_x v_z+v_x v'_x+2\mu_1 R'_{11}+\mu_2 R'_{12}+\mu_3 R'_{13}=0\\
\varepsilon_{R12}:&\ \left(\varepsilon_{R11}-R'_{11}\right)v_y v_x+\left(\varepsilon_{R12}-R'_{12}\right)v_y^2+\left(\varepsilon_{R13}-R'_{13}\right)v_y v_z+v_y v'_x+\mu_2 R'_{11}+2\mu_4 R'_{12}+\mu_5 R'_{13}=0\\
\varepsilon_{R13}:&\ \left(\varepsilon_{R11}-R'_{11}\right)v_z v_x+\left(\varepsilon_{R12}-R'_{12}\right)v_z v_y+\left(\varepsilon_{R13}-R'_{13}\right)v_z^2+v_z v'_x+\mu_3 R'_{11}+\mu_5 R'_{12}+2\mu_6 R'_{13}=0\\
\varepsilon_{R21}:&\ \left(\varepsilon_{R21}-R'_{21}\right)v_x^2+\left(\varepsilon_{R22}-R'_{22}\right)v_x v_y+\left(R'_{23}-\varepsilon_{R23}\right)v_x v_z+v_x v'_y+2\mu_1 R'_{21}+\mu_2 R'_{22}+\mu_3 R'_{23}=0\\
\varepsilon_{R22}:&\ \left(\varepsilon_{R21}-R'_{21}\right)v_y v_x+\left(\varepsilon_{R22}-R'_{22}\right)v_y^2+\left(\varepsilon_{R23}-R'_{23}\right)v_y v_z+v_y v'_y+\mu_2 R'_{21}+2\mu_4 R'_{22}+\mu_5 R'_{23}=0\\
\varepsilon_{R23}:&\ \left(\varepsilon_{R21}-R'_{21}\right)v_z v_x+\left(\varepsilon_{R22}-R'_{22}\right)v_z v_y+\left(\varepsilon_{R23}-R'_{23}\right)v_z^2+v_z v'_y+\mu_3 R'_{21}+\mu_5 R'_{22}+2\mu_6 R'_{23}=0\\
\varepsilon_{R31}:&\ \left(\varepsilon_{R31}-R'_{31}\right)v_x^2+\left(\varepsilon_{R32}-R'_{32}\right)v_x v_y+\left(\varepsilon_{R33}-R'_{33}\right)v_x v_z+v_x v'_z+2\mu_1 R'_{31}+\mu_2 R'_{32}+\mu_3 R'_{33}=0\\
\varepsilon_{R32}:&\ \left(\varepsilon_{R31}-R'_{31}\right)v_y v_x+\left(\varepsilon_{R32}-R'_{32}\right)v_y^2+\left(\varepsilon_{R33}-R'_{33}\right)v_y v_z+v_y v'_z+\mu_2 R'_{31}+2\mu_4 R'_{32}+\mu_5 R'_{33}=0\\
\varepsilon_{R33}:&\ \left(\varepsilon_{R31}-R'_{31}\right)v_z v_x+\left(\varepsilon_{R32}-R'_{32}\right)v_z v_y+\left(\varepsilon_{R33}-R'_{33}\right)v_z^2+v_z v'_z+\mu_3 R'_{31}+\mu_5 R'_{32}+2\mu_6 R'_{33}=0
\end{aligned}
$$

（12）

（9）も合わせて表すと：

$$
\begin{pmatrix}
v_x^2 & v_x v_y & v_x v_z & 0 & 0 & 0 & 0 & 0 & 0 & 2R'_{11} & R'_{12} & R'_{13} & 0 & 0 & 0 \\
v_y v_x & v_y^2 & v_y v_z & 0 & 0 & 0 & 0 & 0 & 0 & 0 & R'_{11} & 0 & 2R'_{12} & R'_{13} & 0 \\
v_z v_x & v_z v_y & v_z^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & R'_{11} & 0 & R'_{12} & 2R'_{13} \\
0 & 0 & 0 & v_x^2 & v_x v_y & v_x v_z & 0 & 0 & 0 & 2R'_{21} & R'_{22} & R'_{23} & 0 & 0 & 0 \\
0 & 0 & 0 & v_y v_x & v_y^2 & v_y v_z & 0 & 0 & 0 & 0 & R'_{21} & 0 & 2R'_{22} & R'_{23} & 0 \\
0 & 0 & 0 & v_z v_x & v_z v_y & v_z^2 & 0 & 0 & 0 & 0 & 0 & R'_{21} & 0 & R'_{22} & 2R'_{23} \\
0 & 0 & 0 & 0 & 0 & 0 & v_x^2 & v_x v_y & v_x v_z & 2R'_{31} & R'_{32} & R'_{33} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & v_y v_x & v_y^2 & v_y v_z & 0 & R'_{31} & 0 & 2R'_{32} & R'_{33} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & v_z v_x & v_z v_y & v_z^2 & 0 & 0 & R'_{31} & 0 & R'_{32} & 2R'_{33} \\
2R'_{11} & 0 & 0 & 2R'_{21} & 0 & 0 & 2R'_{31} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
R'_{12} & R'_{11} & 0 & R'_{22} & R'_{21} & 0 & R'_{32} & R'_{31} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
R'_{13} & 0 & R'_{11} & R'_{23} & 0 & R'_{21} & R'_{33} & 0 & R'_{31} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2R'_{12} & 0 & 0 & 2R'_{22} & 0 & 0 & 2R'_{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & R'_{13} & R'_{12} & 0 & R'_{23} & R'_{22} & 0 & R'_{33} & R'_{32} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2R'_{13} & 0 & 0 & 2R'_{23} & 0 & 0 & 2R'_{33} & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

$$
\begin{pmatrix}
\varepsilon_{R11} \\ \varepsilon_{R12} \\ \varepsilon_{R13} \\ \varepsilon_{R21} \\ \varepsilon_{R22} \\ \varepsilon_{R23} \\ \varepsilon_{R31} \\ \varepsilon_{R32} \\ \varepsilon_{R33} \\ \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \\ \mu_5 \\ \mu_6
\end{pmatrix}
=
\begin{pmatrix}
R'_{11} v_x^2 + R'_{12} v_x v_y + R'_{13} v_x v_z - v_x v'_x \\
R'_{11} v_y v_x + R'_{12} v_y^2 + R'_{13} v_y v_z - v_y v'_x \\
R'_{11} v_z v_x + R'_{12} v_z v_y + R'_{13} v_z^2 - v_z v'_x \\
R'_{21} v_x^2 + R'_{22} v_x v_y + R'_{23} v_x v_z - v_x v'_y \\
R'_{21} v_y v_x + R'_{22} v_y^2 + R'_{23} v_y v_z - v_y v'_y \\
R'_{21} v_z v_x + R'_{22} v_z v_y + R'_{23} v_z^2 - v_z v'_y \\
R'_{31} v_x^2 + R'_{32} v_x v_y + R'_{33} v_x v_z - v_x v'_z \\
R'_{31} v_y v_x + R'_{32} v_y^2 + R'_{33} v_y v_z - v_y v'_z \\
R'_{31} v_z v_x + R'_{32} v_z v_y + R'_{33} v_z^2 - v_z v'_z \\
\phi'_1 \\ \phi'_2 \\ \phi'_3 \\ \phi'_4 \\ \phi'_5 \\ \phi'_6
\end{pmatrix}
$$

(13)

ただし、v、v'については$\sum_{i=1}^{N} v_i$ 等と読み替えるものとする（例$v_x v_y$は$\sum_{i=1}^{N} v_{ix} v_{iy}$）。

計算法

$(v_i\ v'_i)\{1,2,\ \ \ ,N\}$を入力し、

$$\begin{pmatrix} \sum_{i=1}^{N} v_x^2 & \sum_{i=1}^{N} v_x v_y & \sum_{i=1}^{N} v_x v_z \\ \sum_{i=1}^{N} v_y v_x & \sum_{i=1}^{N} v_y^2 & \sum_{i=1}^{N} v_y v_z \\ \sum_{i=1}^{N} v_z v_x & \sum_{i=1}^{N} v_z v_y & \sum_{i=1}^{N} v_z^2 \end{pmatrix}$$ および

$$\begin{pmatrix} \sum_{i=1}^{N} v_x v'_x & \sum_{i=1}^{N} v_x v'_y & \sum_{i=1}^{N} v_x v'_z \\ \sum_{i=1}^{N} v_y v'_x & \sum_{i=1}^{N} v_y v'_y & \sum_{i=1}^{N} v_y v'_z \\ \sum_{i=1}^{N} v_z v'_x & \sum_{i=1}^{N} v_z v'_y & \sum_{i=1}^{N} v_z v'_z \end{pmatrix}$$ の各項に相当する積和を求めておく。

R'の初期値を定める（初期値の例：　$R' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$）。$R'_{11}$～$R'_{33}$を（7）の$\phi_1$～$\phi_6$に代入して$\phi_1$～$\phi_6$の推定値$\phi'_1$～$\phi'_6$を求める。これで（12）のすべての要素が計算できる。これをたとえば掃き出し法で解くとε_Rが求まる（形式上μも求まるが特に使わない）。

　ε_Rが求まったら、これから$R = R' - \varepsilon_R$によりRを推定する。すなわち仮に推定したR'によってε_Rが求まりRが推定できる。これにより得たRをR'として次回のRを推定する。これをくり返して収束が得られる（たとえばϕ'_1～ϕ'_6が0に近づく）まで続ける（Newton法の考えを使っている）。一定回数以上くり返して収束しないときは打ち切る。

　Rが求まったら、回転後の x, y, z軸を$\begin{pmatrix} R_{11} \\ R_{21} \\ R_{31} \end{pmatrix}$、$\begin{pmatrix} R_{12} \\ R_{22} \\ R_{32} \end{pmatrix}$、$\begin{pmatrix} R_{13} \\ R_{23} \\ R_{33} \end{pmatrix}$によって求め、$R\boldsymbol{v}_i$を

$\boldsymbol{v}_i^t \begin{pmatrix} R_{11} \\ R_{21} \\ R_{31} \end{pmatrix}$、$\boldsymbol{v}_i^t \begin{pmatrix} R_{12} \\ R_{22} \\ R_{32} \end{pmatrix}$、$\boldsymbol{v}_i^t \begin{pmatrix} R_{13} \\ R_{23} \\ R_{33} \end{pmatrix}$で求めて（観測データと重ねて）表示する。

参考文献

[1]大島正毅, 白井良明： 3 次元情報を用いた物体認識, 電子通信学会論文誌, vol. J65-D, No. 5, pp. 30-35(1982).

以上

付録　ベクトル群の重ね合わせプログラム

メインプログラム（テスト用）rotmtst.c　（titan:~ oshima/Research_pro/rotation）

```
/*   --- rotmtst.c
     --- prog. to test rotation matching
     --- coded by M. Oshima 2001.03.03
     --- last modification  2001.03.06 */

#include "rotmain.h"

  struct dbflag dbflag = {1, 0, 0, 0};

/* --- usage rotmtst matdata */
int
main(int argc, char *argv[]){
  char data_name[100];
  FILE *fpin;
  int rlt, i, x, y, k;
  struct vpr *vpr;
  double r[3][3];
  double t1, t2, t3, t4, t5, t6;
  int nofv; /* number of v */ int nofvlim = 100;

  if( argc < 2 ){
    printf("Usage: [Program][data_name] \ n");
    return (int)NULL;
  }
  sprintf(data_name,"%s",argv[1]); /*データ名の設定*/
  fpin = fopen( data_name,"r" );
  if( fpin==NULL ){
    printf("Can't open %s.Abort!! \ n",*data_name);
    return NULL;
  }
  vpr = (struct vpr *)malloc(nofvlim * sizeof(struct vpr *));
  i = 0;
  while((rlt = fscanf(fpin,
            "%lf %lf %lf %lf %lf %lf", &t1, &t2, &t3, &t4, &t5, &t6)) != EOF)
{
    (vpr + i)->vmx = t1; (vpr + i)->vmy = t2; (vpr + i)->vmz = t3;
    (vpr + i)->vox = t4; (vpr + i)->voy = t5; (vpr + i)->voz = t6;
    i++;
```

```
    if( i > nofvlim){
      printf("supposed size of vector pair is over! \ n");
      return NULL;
    }
  }
  nofv = i;
  rotmat(vpr, nofv, r);
  if(dbflag.fl1 == 1){
    printf("----- The answer \ n");
    for(y = 0; y < 3; y++){
      for(x = 0; x < 3; x++){
        printf("%10lf",r[y][x]);
      }
      printf(" \ n");
    }
  }

}
```

マッチング関連の関数　rot.c （titan:~ oshima/Research_pro/rotation）

```
/* rot.c
  --- coded by M. Oshima 2001.03.04
  --- last modification  2001.03.20 */


#include "rotsub.h"

/* ***************** */
int
matrx(double *a, int m, int n){
/* ***************** */
/*   --- matrx
        to solve linear simultaneous equations by sweep out method
     --- original coding by Miyamoto with FORTRAN
     --- revised and ported to C by M. Oshima */
  extern struct dbflag dbflag;
  int k, y, x, ymax;
  double t, pvmax, pvll = 1.0e-9, *akx, *aymx;

  for(k=0; k< m; k++){  /* k: number of the pivot */
```

```
/* --- find the best row for the pivot */
   pvmax = 0.0;
   ymax  = 0;
   for(y = k; y < m; y++){
     if( (t = fabs( *(a + y * n + k)) ) > pvmax ){  /* a[y][k] */
       pvmax = t;
       ymax  = y;
     }
   }
   if(dbflag.fl4 == 1) printf("pvmax, ymax = %lf %d \ n",pvmax, ymax);
   if(pvmax < pvll ){
     if(dbflag.fl1 == 1){
       printf(
      "lowest limit of pivot exeeded in matrx  \ n k,ymax,pvmax= %d %d %lf \ n"
        , k,ymax,pvmax);
       getchar();
     }
     return NULL;
   }
/*  --- exchange columm */
   if(ymax != k){
     for(x = k; x < n; x++){
       akx = a + k *  n + x;
       t = *akx; /* a[k][x] */
       aymx = a + ymax * n + x;
       *akx = *aymx; /* a[k][[x] = a[ymax][[x] */
       *aymx = t; /* a[ymax][x] */
     }
   } /* --- end of pivot change */
   /* --- sweep out */
   /* normalize the row */
   t = *(a + k * n + k); /* a[k][k] */
   for(x = k; x < n; x++){
     *(a + k * n + x) /=  t; /* a[k][x] */
   }
/* --- subtract for sweep out */
   for(y = 0; y < m; y++){
     if( y != k){
       t = *(a + y * n + k); /* a[y][k] */
       for(x = k; x < n; x++){
         *(a + y * n + x) -= t * *(a + k * n + x);
```

```
      } /* a[y][x]   =       a[y][x]     - t *  a[k][x]  */
     }
    } /* --- y loop end */
    if(dbflag.fl4 == 1){
     printf(" \ n--- k = %d  \ n", k);
     for(y = 0; y < m; y++){
      for(x = 0; x < n; x++){
       printf("%10lf ", *(a + y * n + x));
       if(x == n-1 ¦¦ x % 20 == 19) printf(" \ n");
      }
     }
    }
  }/* --- k loop end */
}

/*****************************/
double *
rotmat(struct vpr *vpr, int nofv, double rc[][3]){
/*****************************/
/* 与えられた法線ペア（vpr）からvmvm, vmvoを計算し、r, fiの現在推測
値（rc, fic）から線形連立方程式（aで代表される）を解きrを求める。これを
繰り返し誤差erが全要素について0に近ければ終了する。 */

 extern struct dbflag dbflag;
 double er[3][3], a[15][16];
 double vm[3], vo[3], vmvm[3][3], vmvo[3][3], vovo[3], t1, t2, fic[6]
   , erl = 1.0E-3, e, m[6], g;
 int i, y, x, nofc, k, try, trymax = 100;

 for(y = 0; y < 3; y++){
  for(x = 0; x < 3; x++){
   vmvm[y][x] = 0.0; vmvo[y][x] = 0.0;
  }
  vm[y] = vo[y] = vovo[y] = 0.0;
 }

 for(i = 0; i < nofv; i++){
  for(y = 0; y < 3; y++){
   if( y == 0) {t1 = (vpr + i)->vmx; t2 = (vpr + i)->vox; }
   if( y == 1) {t1 = (vpr + i)->vmy; t2 = (vpr + i)->voy; }
   if( y == 2) {t1 = (vpr + i)->vmz; t2 = (vpr + i)->voz; }
```

```
    vm[y] += t1; vo[y] += t2; vovo[y] += t2 * t2;
    vmvm[y][0] += t1 * (vpr + i)->vmx;
    vmvm[y][1] += t1 * (vpr + i)->vmy;
    vmvm[y][2] += t1 * (vpr + i)->vmz;
    vmvo[y][0] += t1 * (vpr + i)->vox;
    vmvo[y][1] += t1 * (vpr + i)->voy;
    vmvo[y][2] += t1 * (vpr + i)->voz;
   }
  }
/* --- set initial rc */
  rc[0][1] = rc[0][2] = rc[1][0] = rc[1][2] = rc[2][0] = rc[2][1] = 0.0;
  rc[0][0] = rc[1][1] = rc[2][2] = 1.0;
  calfic(rc, fic);

  try =0;
  if(dbflag.fl1 == 1){
   printf("try, rc = %d  \ n",try);
   for(y = 0; y < 3; y++){
    for(x = 0; x < 3; x++){
     printf("%10lf",rc[y][x]);
    }
    printf(" \ n");
   }
  }

  do{                /* ----- itterative approximation of r */
   try++;
   calca(vmvm, vmvo, rc, fic, a);
   matrx(&a[0][0], 15, 16);   /* ----- solve simultaneous equations to find r *
/

   k = 0; nofc =0; /* number of converge */
   for(y = 0; y < 3; y++){
    for(x = 0; x < 3; x++){
     er[y][x] = a[k++][15];
     rc[y][x] -= er[y][x];
      if(fabs(er[y][x]) < erl) nofc++;
    }
   }
   calfic(rc, fic);
   if(dbflag.fl1 == 1){
```

```
    printf("try, nofc, er, rc = %d %d \ n",try, nofc);
    for(k = 0; k < 2; k++){
     for(y = 0; y < 3; y++){
      for(x = 0; x < 3; x++){
       if(k == 0) printf("%10lf",er[y][x]);
       if(k == 1) printf("%10lf",rc[y][x]);
      }
      printf(" \ n");
     }
     printf(" \ n");
     for(y = 0; y< 3; y++){
      e += rc[y][0] * vmvm[0][0] + rc[y][1] * vmvm[1][1]
        + rc[y][2] * vmvm[2][2] + vovo[y]
          + 2 * ( rc[y][0] * rc[y][1] *vm[0] * vm[1]
                + rc[y][0] * rc[y][2] *vm[0] * vm[2]
                + rc[y][1] * rc[y][2] *vm[1] * vm[2] )
           - 2 * ( rc[y][0] * vm[0] + rc[y][1] * vm[1] + rc[y][2] * vm[2])
              * vo[y] ;
     }
     printf("e = %10lf \ n",e);
     printf("m =  \ n");
     for(y = 0; y < 6; y++){
      m[y] = a[y + 9][15]; /* g -= m[y] * */
      printf("%10lf",m[y]);
     }
     printf(" \ n");
     g = - 0.5 * e;
     for(i = 0; i < 6; i++){
      g += m[i] * fic[i];
     }
     printf("g= %lf \ n",g);
    }
   }
  } while(nofc < 9 && try < trymax);

}

/**********************/
double *
calfic( double r[][3], double *fic){
/**********************/
```

```
  fic[0] = r[0][0] * r[0][0] + r[1][0] * r[1][0] + r[2][0] * r[2][0] - 1.0;
  fic[1] = r[0][0] * r[0][1] + r[1][0] * r[1][1] + r[2][0] * r[2][1]      ;
  fic[2] = r[0][0] * r[0][2] + r[1][0] * r[1][2] + r[2][0] * r[2][2]      ;
  fic[3] = r[0][1] * r[0][1] + r[1][1] * r[1][1] + r[2][1] * r[2][1] - 1.0;
  fic[4] = r[0][1] * r[0][2] + r[1][1] * r[1][2] + r[2][1] * r[2][2]      ;
  fic[5] = r[0][2] * r[0][2] + r[1][2] * r[1][2] + r[2][2] * r[2][2] - 1.0;
  if(dbflag.fl4 == 1){
    printf("fic = \ n %lf %lf %lf %lf %lf %lf \ n",fic[0],fic[1],fic[2],fic[3],
          fic[4],fic[5]);
  }
}

/*************************************************************************
*******/
double *
calca(double vmvm[][3], double vmvo[][3], double rc[][3], double *fic, double a[
][16]){
/*************************************************************************
*******/
  int y, x, k;

  for(y = 0; y< 15; y++){
    for(x = 0; x < 16; x++){
      a[y][x] = 0.0;
    }
  }
  for(k = 0; k < 9; k += 3){
    for(y = 0; y < 3; y++){
      for(x = 0; x < 3; x++){
        a[k+y][k+x] = vmvm[y][x];
      }
    }
  }
  a[ 0][ 9] = 2.0 * rc[0][0]; a[ 0][10] = rc[0][1]; a[ 0][11] = rc[0][2];
  a[ 1][10] = rc[0][0]; a[ 1][12] = 2.0 * rc[0][1]; a[ 1][13] = rc[0][2];
  a[ 2][11] = rc[0][0]; a[ 2][13] = rc[0][1]; a[ 2][14] = 2.0 * rc[0][2];
  a[ 3][ 9] = 2.0 * rc[1][0]; a[ 3][10] = rc[1][1]; a[ 3][11] = rc[1][2];
  a[ 4][10] = rc[1][0]; a[ 4][12] = 2.0 * rc[1][1]; a[ 4][13] = rc[1][2];
  a[ 5][11] = rc[1][0]; a[ 5][13] = rc[1][1]; a[ 5][14] = 2.0 * rc[1][2];
  a[ 6][ 9] = 2.0 * rc[2][0]; a[ 6][10] = rc[2][1]; a[ 6][11] = rc[2][2];
```

```
  a[ 7][10] = rc[2][0]; a[ 7][12] = 2.0 * rc[2][1]; a[ 7][13] = rc[2][2];
  a[ 8][11] = rc[2][0]; a[ 8][13] = rc[2][1]; a[ 8][14] = 2.0 * rc[2][2];
  a[ 9][ 0] = 2.0 * rc[0][0]; a[ 9][ 3] = 2.0 * rc[1][0]; a[ 9][ 6] = 2.0 * rc[2
][0];
  a[10][ 0] = rc[0][1]; a[10][ 1] = rc[0][0]; a[10][ 3] = rc[1][1];
  a[10][ 4] = rc[1][0]; a[10][ 6] = rc[2][1]; a[10][ 7] = rc[2][0];
  a[11][ 0] = rc[0][2]; a[11][ 2] = rc[0][0]; a[11][ 3] = rc[1][2];
  a[11][ 5] = rc[1][0]; a[11][ 6] = rc[2][2]; a[11][ 8] = rc[2][0];
  a[12][ 1] = 2.0 * rc[0][1]; a[12][ 4] = 2.0 * rc[1][1]; a[12][ 7] = 2.0 * rc[2
][1];
  a[13][ 1] = rc[0][2]; a[13][ 2] = rc[0][1]; a[13][ 4] = rc[1][2];
  a[13][ 5] = rc[1][1]; a[13][ 7] = rc[2][2]; a[13][ 8] = rc[2][1];
  a[14][ 2] = 2.0 * rc[0][2]; a[14][ 5] = 2.0 * rc[1][2]; a[14][ 8] = 2.0 * rc[2
][2];
  a[ 0][15] = rc[0][0] * vmvm[0][0] + rc[0][1] * vmvm[0][1]
          + rc[0][2] * vmvm[0][2] - vmvo[0][0];
  a[ 1][15] = rc[0][0] * vmvm[1][0] + rc[0][1] * vmvm[1][1]
          + rc[0][2] * vmvm[1][2] - vmvo[1][0];
  a[ 2][15] = rc[0][0] * vmvm[2][0] + rc[0][1] * vmvm[2][1]
          + rc[0][2] * vmvm[2][2] - vmvo[2][0];
  a[ 3][15] = rc[1][0] * vmvm[0][0] + rc[1][1] * vmvm[0][1]
          + rc[1][2] * vmvm[0][2] - vmvo[0][1];
  a[ 4][15] = rc[1][0] * vmvm[1][0] + rc[1][1] * vmvm[1][1]
          + rc[1][2] * vmvm[1][2] - vmvo[1][1];
  a[ 5][15] = rc[1][0] * vmvm[2][0] + rc[1][1] * vmvm[2][1]
          + rc[1][2] * vmvm[2][2] - vmvo[2][1];
  a[ 6][15] = rc[2][0] * vmvm[0][0] + rc[2][1] * vmvm[0][1]
          + rc[2][2] * vmvm[0][2] - vmvo[0][2];
  a[ 7][15] = rc[2][0] * vmvm[1][0] + rc[2][1] * vmvm[1][1]
          + rc[2][2] * vmvm[1][2] - vmvo[1][2];
  a[ 8][15] = rc[2][0] * vmvm[2][0] + rc[2][1] * vmvm[2][1]
          + rc[2][2] * vmvm[2][2] - vmvo[2][2];
  for(y = 0; y < 6; y++){
    a[9 + y][15] = fic[y];
  }

}
```

ヘッダファイル rotmain.h （メインプログラムからincludeするべきもの）

```
/* --- rotmain.h ver1.0 */
/*     ----- last modification 2001.03.04 */
/* rotmain.h は回転マッチング関係のmainとなるソースからincludeする
べきもの。必要な共通的なものがこれで自動的に定義される */


#include "rot.h"

/*Window              root;
Display             *disp;
Visual              *vis;
GC                  gc; */
/*Colormap            cmap;*/
/*XSetWindowAttributes  atr;*/
/*FILE                *fpin; */
/*XImage              *image;*/
/*struct                rasterfile          raster;*/
```

ヘッダファイル rotsub.h （サブプログラムからincludeするべきもの）

```
/* --- rotsub.h ver1.0 */

/* rotsub.h は回転マッチング関係のsubとなるソースからincludeする
べきもの。必要な共通的なものがこれで自動的に定義される */

#include "rot.h"
/*   -----                  last modification 2001.03.04 */
/*extern      Window              root;
extern      Display             *disp;
extern      Visual              *vis;
extern      GC                  gc; */
/*extern          Colormap            cmap;*/
/*extern          XSetWindowAttributes    atr;2000.08.07*/
/*extern      FILE                *fpin;*/
/*extern      XImage              *image;*/
/*extern      struct                rasterfile   raster;*/
```

共通ヘッダファイル　rot.h （titan:~ oshima/Research_pro/rotation）

```
/* --- rot.h ver1.0 */
/* --- coded by Masaki Oshima  1st release(ver1.0)    2001.03.04  */
/*                          last modification      2001.03.04  */

/* rot.h は回転マッチングに必要な共通的なものを定義する    */

/*   -------- 共通にincludeするべきもの -------- */
#include <stdio.h>
#include <math.h>

/*   -------- 共通的なdefine文 -------- */

/*   -------- 共通的な変数文 -------- */
struct dbflag{
  int fl1 : 1; /* highest priority debug message */
  int fl2 : 1;
  int fl3 : 1;
  int fl4 : 1;
};

/*   -------- 共通的構造体の定義 -------- */
/* --- vector pair for rotation matching */
struct vpr{double vmx, vmy, vmz, vox, voy, voz;};

/*   -------- 共通マクロ定義 -------- */

/*   -------- 関数プロトタイプ -------- */
int
matrx(double *a, int m, int n);

double *
rotmat(struct vpr *vpr, int nofv, double[][3]);

double *
calfic( double r[][3], double *fic);

double *
calca(double vmvm[][3], double vmvo[][3], double rc[][3], double *fic, double[][
16]);
```

Makefile

```
#--- Makefile_titan 2001.03.04
file = rotmtst

CFLAGS=-g
#CPPFLAGS=-I/usr/local/X11R5/include -I/usr/openwin/share/include/pixrect
#LDFLAGS=-L/usr/lib/X11
CC=cc

options = -lm
#options = -lXaw -lXmu -lXt -lXext -lX11 -lXwchar -lm -lsocket -lnsl

$(file) : $(file).o rot.o
	$(CC) $(CFLAGS) -o $(file) $(file).o rot.o $(options)
$(file).o : $(file).c rotmain.h
	$(CC) $(CFLAGS) -c $(file).c
rot.o : rot.c rot.h rotsub.h
	$(CC) $(CPPFLAGS) $(CFLAGS) -c  rot.c
```

```
/*   --- mkrotdt.c
     --- prog. to make rotation matrix data
     --- coded by M. Oshima 2001.03.18
     --- last modification  2001.03.18 */
#include <stdio.h>
#include <math.h>

/* --- usage mkrotdt roti thdt roto */
int
main(int argc, char *argv[]){
  char data_roti[100], data_thdt[100], data_roto[100];
  FILE *fpri, *fpth, *fpro;
  int rlt, i, x, y, axis;
  double ri[3][3], ro[3][3], r[3][3], th;
  int nofv; /* number of v */ int nofvlim = 100;

  if( argc < 2 ){
    printf("Usage: [Program][data_name] \ n");
    return (int)NULL;
  }
```

```
sprintf(data_roti,"%s",argv[1]); /*データ名の設定*/
sprintf(data_thdt,"%s",argv[2]);
sprintf(data_roto,"%s",argv[3]);
fpri = fopen( data_roti,"r" );
fpth = fopen( data_thdt,"r" );
fpro = fopen( data_roto,"w" );
if( fpri == NULL ){
  printf("Can't open %s.Abort!! \ n",*data_roti);
  return NULL;
}
for(y = 0; y < 3; y++){
  fscanf(fpri,"%lf %lf %lf",&ri[y][0], &ri[y][1], &ri[y][2]);
}
printf("----- ri  \ n");
for(y = 0; y < 3; y++){
  for(x = 0; x < 3; x++){
    printf("%10lf",ri[y][x]);
  }
  printf(" \ n");
}
i = 0;
while((rlt = fscanf(fpth, "%d %lf",&axis, &th)) != EOF){
  switch (axis){
  case 0: /* rotate around x axis */
    r[0][0] =     1.0; r[0][1] =     0.0; r[0][2] =     0.0;
    r[1][0] =     0.0; r[1][1] =  cos(th); r[1][2] = -sin(th);
    r[2][0] =     0.0; r[2][1] =  sin(th); r[2][2] =  cos(th);
    break;
  case 1: /* rotate around y axis */
    r[0][0] =  cos(th); r[0][1] =     0.0; r[0][2] =  sin(th);
    r[1][0] =     0.0; r[1][1] =     1.0; r[1][2] = 0.0;
    r[2][0] = -sin(th); r[2][1] =     0.0; r[2][2] =  cos(th);
    break;
  case 2: /* rotate around z axis */
    r[0][0] =  cos(th); r[0][1] = -sin(th); r[0][2] = 0.0;
    r[1][0] =  sin(th); r[1][1] =  cos(th); r[1][2] = 0.0;
    r[2][0] =     0.0; r[2][1] =     0.0; r[2][2] = 1.0;
    break;
  default: break;
  }
  for(y = 0; y < 3; y++){
```

```
      for(x = 0; x < 3; x++){
        ro[y][x]
          = r[y][0] * ri[0][x] + r[y][1] * ri[1][x] + r[y][2] * ri[2][x];
      }
    }
    i++;
    if( i > nofvlim){
      printf("supposed size of vector pair is over! \ n");
      return NULL;
    }
    for(y = 0; y < 3; y++){
      fprintf(fpro,"%10lf %10lf %10lf \ n",ro[y][0], ro[y][1], ro[y][2]);
    }
  }
  nofv = i;
  printf("----- ro  \ n");
  for(y = 0; y < 3; y++){
    for(x = 0; x < 3; x++){
      printf("%10lf",ro[y][x]);
    }
    printf(" \ n");
  }
}
```

回転マトリックス作成ルーチン　mkrotdt.c

```
/*   --- mkrotdt.c
     --- prog. to make rotation matrix data
     --- coded by M. Oshima 2001.03.18
     --- last modification  2001.03.18 */
#include <stdio.h>
#include <math.h>

/* --- usage mkrotdt roti thdt roto */
int
main(int argc, char *argv[]){
  char data_roti[100], data_thdt[100], data_roto[100];
  FILE *fpri, *fpth, *fpro;
  int rlt, i, x, y, axis;
  double ri[3][3], ro[3][3], r[3][3], th;
```

```
int nofv; /* number of v */ int nofvlim = 100;

if( argc < 2 ){
  printf("Usage: [Program][data_name] \ n");
  return (int)NULL;
}
sprintf(data_roti,"%s",argv[1]); /*データ名の設定*/
sprintf(data_thdt,"%s",argv[2]);
sprintf(data_roto,"%s",argv[3]);
fpri = fopen( data_roti,"r" );
fpth = fopen( data_thdt,"r" );
fpro = fopen( data_roto,"w" );
if( fpri == NULL ){
  printf("Can't open %s.Abort!! \ n",*data_roti);
  return NULL;
}
for(y = 0; y < 3; y++){
  fscanf(fpri,"%lf %lf %lf",&ri[y][0], &ri[y][1], &ri[y][2]);
}
printf("----- ri  \ n");
for(y = 0; y < 3; y++){
  for(x = 0; x < 3; x++){
    printf("%10lf",ri[y][x]);
  }
  printf(" \ n");
}
i = 0;
while((rlt = fscanf(fpth, "%d %lf",&axis, &th)) != EOF){
  switch (axis){
  case 0: /* rotate around x axis */
    r[0][0] =     1.0; r[0][1] =     0.0; r[0][2] =     0.0;
    r[1][0] =     0.0; r[1][1] =  cos(th); r[1][2] = -sin(th);
    r[2][0] =     0.0; r[2][1] =  sin(th); r[2][2] =  cos(th);
    break;
  case 1: /* rotate around y axis */
    r[0][0] =  cos(th); r[0][1] =     0.0; r[0][2] =  sin(th);
    r[1][0] =     0.0; r[1][1] =     1.0; r[1][2] = 0.0;
    r[2][0] = -sin(th); r[2][1] =     0.0; r[2][2] =  cos(th);
    break;
  case 2: /* rotate around z axis */
    r[0][0] =  cos(th); r[0][1] = -sin(th); r[0][2] = 0.0;
```

```
      r[1][0] =  sin(th); r[1][1] =  cos(th); r[1][2] = 0.0;
      r[2][0] =      0.0; r[2][1] =      0.0; r[2][2] = 1.0;
      break;
    default: break;
    }
    for(y = 0; y < 3; y++){
      for(x = 0; x < 3; x++){
        ro[y][x]
          = r[y][0] * ri[0][x] + r[y][1] * ri[1][x] + r[y][2] * ri[2][x];
      }
    }
    i++;
    if( i > nofvlim){
      printf("supposed size of vector pair is over! \ n");
      return NULL;
    }
    for(y = 0; y < 3; y++){
      fprintf(fpro,"%10lf %10lf %10lf \ n",ro[y][0], ro[y][1], ro[y][2]);
    }
  }
  nofv = i;
  printf("----- ro  \ n");
  for(y = 0; y < 3; y++){
    for(x = 0; x < 3; x++){
      printf("%10lf",ro[y][x]);
    }
    printf(" \ n");
  }
}
```

上記の入出力roti thdt rotoの例

```
more rot0
1.0 0.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0

more thdt
2 0.5236

more roto
```

```
 0.866025  -0.500001   0.000000
 0.500001   0.866025   0.000000
 0.000000   0.000000   1.000000
```

テスト用メインルーチンrotmtst.cに食わせるデータ（mato）を生成するためのルーチン
mkmtdata.c

```
/*   --- mkmtdata.c
     --- prog. to make data for rotation matching
     --- coded by M. Oshima 2001.03.18
     --- last modification  2001.03.18 */

/* --- usage mkmtdata rot mati mato */

#include <stdio.h>

int
main(int argc, char *argv[]){
  char data_rot[100], data_namei[100], data_nameo[100];
  FILE *fpr, *fpi, *fpo;
  int rlt, i, x, y;
  double xi, yi, zi, xo, yo, zo;
  struct vpr *vpr;
  double r[3][3];
  int nofv; /* number of v */ int nofvlim = 100;

  if( argc < 2 ){
    printf("Usage: [Program][data_name] \ n");
    return (int)NULL;
  }
  sprintf(data_rot  ,"%s",argv[1]); /*データ名の設定*/
  sprintf(data_namei,"%s",argv[2]);
  sprintf(data_nameo,"%s",argv[3]);
  fpr = fopen( data_rot  ,"r" );
  fpi = fopen( data_namei,"r" );
  fpo = fopen( data_nameo,"w" );
  if( fpr == NULL ){
    printf("Can't open %s.Abort!! \ n",*data_rot);
    return NULL;
  }
  for(y =0; y < 3; y++){
```

```
    fscanf(fpr,"%lf %lf %lf",&r[y][0], &r[y][1], &r[y][2]);
  }
  printf("----- r  \ n");
  for(y = 0; y < 3; y++){
    for(x = 0; x < 3; x++){
      printf("%10lf",r[y][x]);
    }
    printf(" \ n");
  }
  i = 0;
  while((rlt = fscanf(fpi,
                "%lf %lf %lf", &xi, &yi, &zi)) != EOF){
    xo = r[0][0] * xi +  r[0][1] * yi + r[0][2] * zi;
    yo = r[1][0] * xi +  r[1][1] * yi + r[1][2] * zi;
    zo = r[2][0] * xi +  r[2][1] * yi + r[2][2] * zi;
    printf("xi, yi, zi, xo, yo, zo = %10lf %10lf %10lf %10lf %10lf %10lf \ n",xi,
yi, zi, xo, yo, zo);
    i++;
    if( i > nofvlim){
      printf("supposed size of vector pair is over! \ n");
      return NULL;
    }
    fprintf(fpo,"%10lf %10lf %10lf %10lf %10lf %10lf \ n",xi, yi, zi, xo, yo, zo);
  }
  nofv = i;
}
```

上記の入出力例　rot mati mato

```
more roto
  0.866025  -0.500001   0.000000
  0.500001   0.866025   0.000000
  0.000000   0.000000   1.000000

more mati
0.5 0.3 0.8124
0.4 0.6 0.6928
-0.7 0.3 0.648074

more mato
  0.500000   0.300000   0.812400   0.283012   0.509808   0.812400
```

 0.400000 0.600000 0.692800 0.046409 0.719615 0.692800
-0.700000 0.300000 0.648074 -0.756218 -0.090193 0.648074

テストルーチンrotmtstの結果（徐々に収束して行く様子が分かる）
初期値は
 1.000000 0.000000 0.000000
 0.000000 1.000000 0.000000
 0.000000 0.000000 1.000000
で与えたものは
0.5 0.3 0.8124
0.4 0.6 0.6928
-0.7 0.3 0.648074
に回転
 0.866025 -0.500001 0.000000
 0.500001 0.866025 0.000000
 0.000000 0.000000 1.000000
を施したものなので、最終結果が
 0.866025 -0.500001 0.000000
 0.500001 0.866025 0.000000
 0.000000 0.000000 1.000000
となることが期待される。

rotmtst mato
try, rc = 0
 1.000000 0.000000 0.000000
 0.000000 1.000000 0.000000
 0.000000 0.000000 1.000000
try, nofc, er, rc = 1 3
 0.000000 0.504553 0.006950
-0.504553 0.000000 0.054375
-0.006950 -0.054375 0.000000

e = -6.500146
m =
 0.059081 0.015724 0.016042 0.013370 0.030613 0.024011
g= 3.268305
 1.000000 -0.504553 -0.006950
 0.504553 1.000000 -0.054375
 0.006950 0.054375 1.000000

```
e = -13.000292
m =
  0.059081  0.015724  0.016042  0.013370  0.030613  0.024011
g= 6.518378
try, nofc, er, rc = 2 2
  0.126822 -0.000556 -0.008148
  0.000878  0.125454 -0.049452
  0.006602  0.055733 -0.001243

e = -19.278001
m =
  0.003139  0.000558  0.000280  0.000398  0.000344  0.000420
g= 9.639057
  0.873178 -0.503996  0.001198
  0.503675  0.874546 -0.004923
  0.000348 -0.001358  1.001243

e = -25.555709
m =
  0.003139  0.000558  0.000280  0.000398  0.000344  0.000420
g= 12.777912
try, nofc, er, rc = 3 1
  0.007125 -0.003976  0.001193
  0.003659  0.008481 -0.004897
  0.000349 -0.001335  0.001230

e = -31.802110
m =
  0.000014  0.000001 -0.000002  0.000001 -0.000001  0.000001
g= 15.901055
  0.866053 -0.500020  0.000005
  0.500016  0.866066 -0.000026
 -0.000001 -0.000023  1.000013

e = -38.048511
m =
  0.000014  0.000001 -0.000002  0.000001 -0.000001  0.000001
g= 19.024256
try, nofc, er, rc = 4 9
  0.000029 -0.000019  0.000006
  0.000015  0.000041 -0.000026
```

```
 -0.000001 -0.000023  0.000013

e = -44.294683
m =
  0.000000  0.000000  0.000000  0.000000 -0.000000 -0.000000
g= 22.147341
  0.866025 -0.500001 -0.000000
  0.500001  0.866025  0.000000
  0.000000 -0.000000  1.000000

e = -50.540854
m =
  0.000000  0.000000  0.000000  0.000000 -0.000000 -0.000000
g= 25.270427
----- The answer
  0.866025 -0.500001 -0.000000
  0.500001  0.866025  0.000000
  0.000000 -0.000000  1.000000

（期待される通りの結果
  0.866025  -0.500001   0.000000
  0.500001   0.866025    0.000000
  0.000000   0.000000    1.000000
が得られている）
```

以上