```
% ro12.m  30-nov-08
% relative orientation by coplanarity
% nov 2007 r07.m does r/o by collinearity
% copied from ro10.m, this one hard wired for
% pair of nikon e5700 coolpix photos of armstrong hall
% format 2560 x 1920 (1280,960 pp)
% f appx= 2403 ( this disagrees with some earlier bundle block files ???)

ctr_samp=1280;
ctr_line=960;
foc=2403;

[seq,pnam,psx,psy]=textread('left12.txt','%d %s %f %f');
[m,n]=size(seq);
[seq2,pnam2,psx2,psy2]=textread('right12.txt','%d %s %f %f');
npnt=m;
xl=zeros(m,1);
yl=zeros(m,1);
xr=zeros(m,1);
yr=zeros(m,1);
xl0=xl;
yl0=yl;
xr0=xr;
yr0=yr;
for i=1:npnt
  xl(i)=psx(i) - ctr_samp;
  yl(i)= -(psy(i) - ctr_line);
  xr(i)=psx2(i) - ctr_samp;
  yr(i)= -(psy2(i) - ctr_line);
  xl0(i)=xl(i);
  yl0(i)=yl(i);
  xr0(i)=xr(i);
  yr0(i)=yr(i);
  end
x0=0;
y0=0;
% set above
% foc=2917;
% foc=input('enter focal length: ');
io=[x0;y0;foc];
n=npnt*4;
u=5;
n0=5+3*npnt;
r=n-n0;
c=r+u;
A=zeros(c,n);
B=zeros(c,u);
f=zeros(c,1);
W=eye(n);
% initial parameter approx
% bx=100;
by=0;
bz=0;
om=0;
ph=0;
kp=0;
p=[by; bz; om; ph; kp];

iter=1;
keep_going=1;
while(keep_going == 1)
  iter
  colA_idx=1;
```

```
  for i=1:c
    l=[xl(i); yl(i); xr(i); yr(i)];
    l0=[xl0(i); yl0(i); xr0(i); yr0(i)];
    [F,a,b]=lin_coplan(io,p,l0);
    A(i,colA_idx:colA_idx+3)=a;
    B(i,:)=b;
    f(i)=-F - a*(l-l0);
    colA_idx=colA_idx + 4;
    end;
  Q=inv(W);
  Qe=A*Q*A';
  We=inv(Qe);
  N=B'*We*B;
  t=B'*We*f;
  iter
  disp('parameter corrections');
  del=inv(N)*t
  p=p+del;
  k=We*(f - B*del);
  v=Q*A'*k;
  for i=1:npnt
    idx=(i-1)*4 + 1;
    vv=v(idx:idx+3);
    xl0(i)=xl(i) + vv(1);
    yl0(i)=yl(i) + vv(2);
    xr0(i)=xr(i) + vv(3);
    yr0(i)=yr(i) + vv(4);
    end
  if(all(abs(del) < 1.0e-06))
    keep_going=0;
    disp('we converged');
    end
  if(iter > 10)
    keep_going=0;
    disp('not converged in 10 iterations');
    end
  iter=iter+1;
  end
disp('residuals');
v
foc
vtwv=v'*v
disp('parameters by, bz, om, ph, kp');
p
```

```
% lin_coplan.m  5-nov-08
% return linearized coplanarity equation
function [Fcp,a,b]=lin_coplan(io,p,l);
F=coplan(io,p,l);
a=zeros(1,4);
b=zeros(1,5);
dl=0.001;
dp=[0.00001 0.00001 0.00000001 0.00000001 0.00000001];
for i=1:4
  l2=l;
  l2(i)=l2(i) + dl;
  F2=coplan(io,p,l2);
  a(i)=(F2-F)/dl;
  end
for i=1:5
  p2=p;
  p2(i)=p2(i) + dp(i);
  F2=coplan(io,p2,l);
  b(i)=(F2-F)/dp(i);
  end
Fcp=F;
```

```
% coplan.m  5-nov-08
% evaluate the coplanarity condition equation

function Fcp = coplan(io,p,l)
x0=io(1);
y0=io(2);
f=io(3);
by=p(1);
bz=p(2);
om=p(3);
ph=p(4);
kp=p(5);
x1=l(1);
y1=l(2);
x2=l(3);
y2=l(4);
M1=eye(3);
M2=m3(kp)*m2(ph)*m1(om);
bx=100;
a1=M1' * [x1-x0; y1-y0; -f];
a2=M2' * [x2-x0; y2-y0; -f];
Fcp=det( [bx by bz; a1' ; a2' ] );
```

```
% m1.m
function m=m1(th)
m=[1 0 0;0 cos(th) sin(th);0 -sin(th) cos(th)];
```

```
% m2.m
function m=m2(th)
m=[cos(th) 0 -sin(th);0 1 0;sin(th) 0 cos(th)];
```

```
% m3.m
function m=m3(th)
m=[cos(th) sin(th) 0;-sin(th) cos(th) 0;0 0 1];
```

```
ro12
iter =
     1
iter =
     1
parameter corrections
del =
        4.517
       15.967
     -0.01251
      0.12975
    -0.011965
iter =
     2
iter =
     2
parameter corrections
del =
      0.31058
      -4.6798
   -0.0068949
    -0.010165
    0.0080179
iter =
     3
iter =
     3
parameter corrections
del =
     -0.14514
      -0.3439
   0.00027773
   0.00053471
  -0.00046582
iter =
     4
iter =
     4
parameter corrections
del =
     0.002865
    0.0053165
  -4.187e-006
  -7.991e-006
  3.6122e-006
iter =
     5
iter =
     5
parameter corrections
del =
   -0.0002637
   -0.0005396
  4.3731e-007
  8.0648e-007
 -2.5892e-007
iter =
     6
iter =
     6
parameter corrections
del =
  1.0885e-005
```

```
  2.1771e-005
 -1.8165e-008
 -3.2705e-008
  1.0907e-008
iter =
     7
iter =
     7
parameter corrections
del =
 -5.6078e-007
 -8.9465e-007
  1.1244e-009
 -9.7401e-010
 -7.2121e-010
we converged
residuals
v =
     -0.18965
       2.5919
      0.30252
      -2.7833
      0.13731
      -3.1886
     -0.15765
         3.48
    -0.059515
       4.5434
    -0.088567
      -5.1792
     0.014983
      -2.4873
     0.090176
       2.8415
    -0.012941
      0.16122
     0.020685
     -0.16716
     0.042686
     -0.62661
    -0.063514
      0.64801
     0.040236
      -1.2829
    -0.028032
       1.3444
     0.084301
      -1.0877
     -0.13067
       1.1086
     0.080891
       -1.347
      -0.1077
       1.3299
   -0.0071139
      0.16496
    0.0074614
     -0.16424
   -0.0056952
      0.14785
    0.0053978
      -0.1508
     0.014766
```

```
     -0.61605
   -0.0034635
      0.63483
   -0.0045379
     0.092652
    0.0051892
    -0.089939
    -0.075642
       1.9034
     0.070157
      -1.8296
    -0.023474
      0.94547
    0.0068739
     -0.94925
foc =
        2403
vtwv =
        118.7
parameters by, bz, om, ph, kp
p =
        4.685
       10.948
    -0.019131
      0.12011
   -0.0044095
diary off
```

```
% pr8.m  30-nov-08
% copied from pr7.m from d:\classes\ce503_04\ana1
% which has been modified for fall 2007 distance course
% pairwise rectify two images
% hardwired for two nikon e5700 images of armstrong bldg
% for photo1_08 demo project
% some history follows
% this .m file is just a translation of pr.c found in d:\classes\ce503_04\ana1
% which is derived from pairrect.c as distributed with the textbook

% since this comes from coplanarity as implemented in ro12.m
% the left image is identity and origin
% would it be better to split the phi equally between the images ?

% to use for other oriented image pairs, edit UPPER CASE constants:
% enter omega, phi, kappa, XL, YL, ZL left
% enter omega, phi, kappa, XL, YL, ZL right
% enter focal length F (pixels)
% after running, adjust XMIN, YMAX, NOM_PLX to optimize view:
% XMIN: in order to move whole image right, make this smaller (more negative)
% YMAX: in order to move whole image up, make this smaller
% NOM_PLX: locate a bright object, the red component corresponds to the
%     left image. a smaller value of this will move the left (red) image
%     to the right. (this adjustment only for eye comfort - does not
%     affect geometry)

% left arms001.jpg
OMEGA1=   0.000000;
PHI1=     0.000000;
KAPPA1=   0.000;
XL1=      0.000;
YL1=      0.000;
ZL1=      0.000;
F=     2403.000;

% right arms005.jpg
OMEGA2=  -0.019131;
PHI2=     0.12011;
KAPPA2=  -0.0044095;
XL2=      100.000;
YL2=        4.685;
ZL2=       10.948;

% how to set these numbers ???
% can we specify a "null-parallax" point and get the shift constants
% from that ? certainly would be simpler

XMIN= -1550;
YMAX= 1000;
STEP=     1.000;
NOM_PLX= 350;

% inner orientation 6-parameter transformation

x0=0;
y0=0;

% left

a01=y0;
a11=0.0;
a21=-1.0;
b01=x0;
```

```
b11=1.0;
b21=0.0;

% right

a02=y0;
a12=0.0;
a22=-1.0;
b02=x0;
b12=1.0;
b22=0.0;

% running in folder d:\classes\photo1_08\
in_img1=imread('stereo\arms001.jpg');
%image(in_img1);
[INROW1,INCOL1,clr]=size(in_img1);
in_img2=imread('stereo\arms005.jpg');
%image(in_img2);
[INROW2,INCOL2,clr]=size(in_img2);
ROW_SHIFT=fix(INROW2/2);
COL_SHIFT=fix(INCOL2/2);
OUTROW=INROW1;
OUTCOL=INCOL1;
out_img=zeros(OUTROW,OUTCOL,3,'uint8');

M1=m3(KAPPA1)*m2(PHI1)*m1(OMEGA1);
M2=m3(KAPPA2)*m2(PHI2)*m1(OMEGA2);

bx=XL2-XL1;
by=YL2-YL1;
bz=ZL2-ZL1;
thz=atan2(by,bx);
thy=atan2(-bz,sqrt(bx*bx+by*by));
% extract tertiary omegas
p= -asin(M1(1,3));
w1=atan2(M1(2,3)/cos(p),M1(3,3)/cos(p));
p= -asin(M2(1,3));
w2=atan2(M2(2,3)/cos(p),M2(3,3)/cos(p));
thx=(w1+w2)/2.0;

mx=m1(thx);
my=m2(thy);
mz=m3(thz);
mb=mx*my*mz;
mn1=mb*M1';
mn2=mb*M2'

% look out for prior 0->(n-1) range
% now it is 1->n

vecl=zeros(3,1);
vecr=zeros(3,1);
yn=YMAX;
for i=1:OUTROW
  disp('row');
  i
  yn=yn - STEP;
  xn=XMIN;
  for j=1:OUTCOL
    xn=xn+STEP;
    xnl=xn;
    xnr=xnl - (NOM_PLX);
    vecl=[xnl;yn;-F];
```

```
    vecr=[xnr;yn;-F];
    uvw1=mn1'*vecl;
    uvw2=mn2'*vecr;
    xp1=-F*uvw1(1)/uvw1(3);
    yp1=-F*uvw1(2)/uvw1(3);
    xp2=-F*uvw2(1)/uvw2(3);
    yp2=-F*uvw2(2)/uvw2(3);
    row1=a01 + a11*xp1 + a21*yp1 + ROW_SHIFT;
    col1=b01 + b11*xp1 + b21*yp1 + COL_SHIFT;
    row2=a02 + a12*xp2 + a22*yp2 + ROW_SHIFT;
    col2=b02 + b12*xp2 + b22*yp2 + COL_SHIFT;
    % make nearest neighbor interpolation
    irow1=fix(row1 + 0.5);
    icol1=fix(col1 + 0.5);
    irow2=fix(row2 + 0.5);
    icol2=fix(col2 + 0.5);

    % test limits left
    intest=1;
    if((irow1 < 1) | (irow1 > (INROW1)))
      intest=0;
      end
    if((icol1 < 1) | (icol1 > (INCOL1)))
      intest=0;
      end
    if(intest == 1)

intensity=(int16(in_img1(irow1,icol1,1))+int16(in_img1(irow1,icol1,2))+int16(in_img1
(irow1,icol1,3)))/3;
      red=fix(intensity);
    else
      red=150;
      end

    % test limits right
    intest=1;
    if((irow2 < 1) | (irow2 > (INROW2)))
      intest=0;
      end
    if((icol2 < 1) | (icol2 > (INCOL2)))
      intest=0;
      end
    if(intest == 1)

intensity=(int16(in_img2(irow2,icol2,1))+int16(in_img2(irow2,icol2,2))+int16(in_img2
(irow2,icol2,3)))/3;
      green=fix(intensity);
      blue=green;
    else
      green=150;
      blue=green;
      end

    % make the anaglyph stereo output pixel

    out_img(i,j,1)=red;
    out_img(i,j,2)=green;
    out_img(i,j,3)=blue;
    end
  end

image(out_img);
imwrite(out_img,'pr8.jpg','JPEG');
```