Algorithm Description for Image Resection with Physical Sensor Model for Satellite Camera

- Create a function which evaluates 2D misclosure vector given a corresponding image point and GCP coordinates
- Select a subset of the total parameter set
- Read in the image measurements and GCP coordinates
- Read in ephemeris data
- Initialize parameters (usually to zero), place in a parameter vector
- Create another vector with a small, "delta" value for each parameter to use later for numerical derivative
- Begin iteration loop for nonlinear LS resection, this loop should check for convergence and terminate
  - Begin loop cycle over all *rows* of the B-matrix, that is loop over all GCP's
    - Make any needed coordinate conversions to prepare data to pass to your misclosure function
    - Call the function once, to establish the "baseline" or reference value for the misclosure for this point
    - Begin loop cycle over all *columns* of B-matrix, that is a loop over all *selected* parameters
      - For each parameter (column), perturb the corresponding parameter value, send this perturbed parameter vector to the misclosure function
      - Subtract these from the reference values, and divide by deltas to get estimate of the partial derivative
      - Store in appropriate row and column of B
      - Store negative of reference misclosure in the f-vector
      - Continue, next column
    - Continue, next row
  - Having filled in all rows and columns of B and f, now solve the LS problem, note condition number of N

- Add corrections to the parameter vector, check for convergence
- Continue, next iteration

❖ Print summary of results, RMS x, RMS y, parameter values, etc.

See the accompanying file of matlab code that suggests one way of translating that algorithm into a program.

```
% res_templ.m - template for quickbird resect
% show mechanics of
% ** parameter selection
% ** numerical estimation of partial derivatives
% just to show the ideas

NPAR=27;
puse=[18;19;21;22;24;25];
[m,n]=size(puse);
npuse=m;

disp('number of parameters to use');
npuse

use=zeros(NPAR,1);
for i=1:NPAR
  for j=1:npuse
    if puse(j) == i
      use(i)=1;
      end;
    end;
  end;

load qb.inp;
[m,n]=size(qb);
npts=m;
disp('number of data points');
npts

NE=zeros(npuse,npuse);
TE=zeros(npuse,1);
BB=zeros(2*npts,npuse);
FF=zeros(2*npts,1);
delta=zeros(NPAR,1);
par=zeros(NPAR,1);

NLINES=30004;
NSAMPL=27552;

dx0=0;
dx1=0;
dx2=0;
dy0=0;
dy1=0;
dy2=0;
dz0=0;
dz1=0;
dz2=0;
dw0=0;
dw1=0;
dw2=0;
dp0=0;
dp1=0;
dp2=0;
dk0=0;
dk1=0;
dk2=0;

par(1)=Omega;
par(2)=i;
par(3)=w;
par(4)=a;
par(5)=e;
```

```
par(6)=foc;
par(7)=r0;
par(8)=c0;
par(9)=dx0;
par(10)=dx1;
par(11)=dx2;
par(12)=dy0;
par(13)=dy1;
par(14)=dy2;
par(15)=dz0;
par(16)=dz1;
par(17)=dz2;
par(18)=dw0;
par(19)=dw1;
par(20)=dw2;
par(21)=dp0;
par(22)=dp1;
par(23)=dp2;
par(24)=dk0;
par(25)=dk1;
par(26)=dk2;
par(27)=li;

% you might want to change magnitudes here to match units
for i=1:NPAR
  delta(i)=1.0E-06;
  end;

for iter=1:10
  disp('iteration');
  iter
  nxeqn=1;
  nyeqn=2;

  for np=1:npts
    % prepare input data
    phid=qb(np,2) + qb(np,3)/60.0 + qb(np,4)/3600.0;
    phi=phid*(pi/180.0);
    lamd=qb(np,5) + qb(np,6)/60.0 + qb(np,7)/3600.0;
    lambda=lamd*(pi/180.0);
    lambda= -lambda;
    h=qb(np,8);
    line_px=qb(np,9);
    samp_px=qb(np,10);
    usr=gtousr(phi,lambda,h,pare);
    % since pare has a in m, XYZ are in m
    X=usr(1);
    Y=usr(2);
    Z=usr(3);

    % the two elements in the F-vector are the x&y condition equation
    % values, after convergence these are also the residuals

    F=qbceq2(line_px,samp_px,X,Y,Z,par,pare,.....);
    col=1;
    for i=1:NPAR
      if use(i) == 1
        % compute dFx/dp & dFy/dp numerically and use them to
        % fill the coefficient matrix
        pardel=par;
        pardel(i)=pardel(i) + delta(i);
        Fdel=qbceq2(line_px,samp_px,X,Y,Z,pardel,auxpar,pare,times,Mats,Dang);
        dfxdp=(Fdel(1)-F(1))/delta(i);
```

```
        dfydp=(Fdel(2)-F(2))/delta(i);
        BB(nxeqn,col)=dfxdp;
        BB(nyeqn,col)=dfydp;
        col=col+1;
        FF(nxeqn)=-F(1);
        FF(nyeqn)=-F(2);
        end;
      end;
    nxeqn=nxeqn+2;
    nyeqn=nyeqn+2;
    end;

  [nrowB,ncolB]=size(BB);
  NE=BB'*BB;
  TE=BB'*FF;
  cond_N=cond(NE)

  del=inv(NE)*TE;
  disp('parameter corrections');
  del

  col=1;
  for i=1:NPAR
    if use(i) == 1
      par(i)=par(i) + del(col);
      col=col + 1;
      end;
    end;
% next iteration
  end;

% print results
```