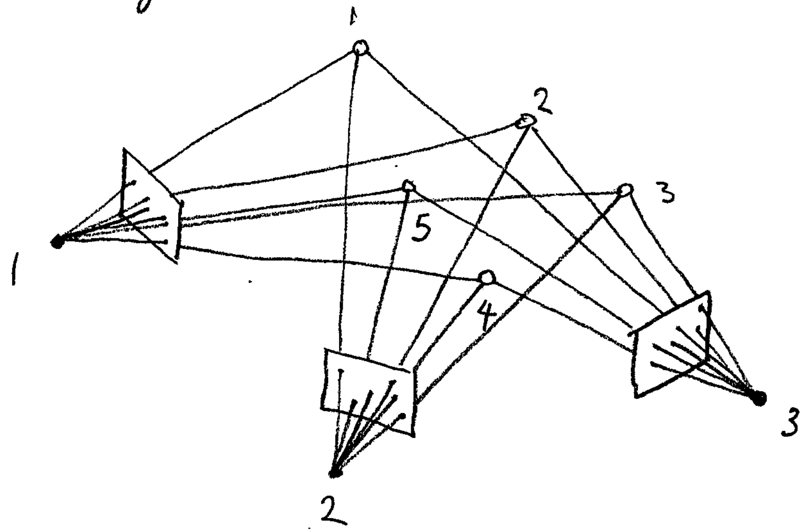
data 2 – Bundle Block Adj & Constraints

assigned Wed 4 Feb 09, due ~~Fri. 13 Feb 09~~ Friday, 20 February



Pho.	Pnt.	x	y
1	1	-14.218	-8.923
1	2	-1.014	-5.172
1	3	8.967	12.493
1	4	9.645	29.359
1	5	2.515	0.835
2	1	-8.043	-3.307
2	2	4.528	-2.073
2	3	9.765	13.540
2	4	-5.737	20.399
2	5	-0.415	2.446
3	1	-10.004	-8.592
3	2	9.345	-7.780
3	3	14.013	19.469
3	4	-21.869	32.213
3	5	-4.235	0.175

$\sigma_x = \sigma_y = .005$

photo approx:

	(1)	(2)	(3)
θ_z	-56°	-23°	0°
θ_x	87°	87°	92°
X	0.4	2.3	4.5
Y	3.9	0.6	2.4
Z	1.5	1.6	1.5

$M = R_1(\theta_x) R_3(\theta_z)$

Write a matlab program to perform the 3D bundle block adjustment for 2 constraint cases:

(1) Fix point 1 @ (3.2, 7.8, 0.4)
Fix point 2 @ (6.1, 7.8, 0.5)
Fix point 3 @ (—, —, 3.5)

(2) inner constraints on all object points

show adjustment results and plot (2D, XY) error ellipses for all object points.

use $x0 = y0 = 0$, $f =$ ~~50.0~~ 35.0

for approximation of object points use "linear" intersection algorithm given on pages 14-10, 14-11 of 2007 Photo 1 notes.

see subsequent page on algorithm revision for your B-matrix, use parameter order:

$| \omega_1 \ \varphi_1 \ \kappa_1 \ X_{L1} \ Y_{L1} \ Z_{L1} | \ \omega_2 \ \varphi_2 \ \kappa_2 \ X_{L2} \ Y_{L2} \ Z_{L2} | \ \omega_3 \ \varphi_3 \ \kappa_3 \ X_{L3} \ Y_{L3} \ Z_{L3} |$

$X_1 \ Y_1 \ Z_1 | \ X_2 \ Y_2 \ Z_2 | \ X_3 \ Y_3 \ Z_3 | \ X_4 \ Y_4 \ Z_4 | \ X_5 \ Y_5 \ Z_5 |$

suggest using matlab "spy" command to confirm structure of filled in B-matrix and N-matrix.

See attached coll.m and lincoll.m to help with creating condition equations & linearization.

Note that N with all unknowns will be rank deficient, so you must incorporate constraints and solve via:

$$\begin{bmatrix} -N & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \Delta \\ k_c \end{bmatrix} = \begin{bmatrix} -t \\ g \end{bmatrix}$$

See attached plot_conf_ell.m and draw_ell.m to help with plotting confidence ellipses in 2D from 2x2 covariance matrix.

50% error ellipse

It was recommended to use the linear intersection equations given in the chapter 14 notes from 2007. Unfortunately those equations fail for the present problem. When you do intersection in object space and you divide by one of the three components of a ray, you must be sure that the chosen component is not near zero. For our terrestrial photogrammetry problem, Z-ZL happens to be near zero for at least on point and this leads to a poor approximation, from which you cannot recover. To fix this we just rederive the linear intersection equations but divide instead by Y-YL which we know for this problem to be comfortably non-zero.

$$\begin{bmatrix} x - x_0 \\ y - y_0 \\ -f \end{bmatrix} = \lambda M \begin{bmatrix} X - X_L \\ Y - Y_L \\ Z - Z_L \end{bmatrix}$$

$$\frac{1}{\lambda} M^T \begin{bmatrix} x - x_0 \\ y - y_0 \\ -f \end{bmatrix} = \begin{bmatrix} X - X_L \\ Y - Y_L \\ Z - Z_L \end{bmatrix}$$

$$\frac{1}{\lambda} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} X - X_L \\ Y - Y_L \\ Z - Z_L \end{bmatrix}$$

$$c_1 = \frac{u}{v} = \frac{X - X_L}{Y - Y_L}$$

$$c_2 = \frac{w}{v} = \frac{Z - Z_L}{Y - Y_L}$$

Rearrange this as a matrix equation with a coefficient matrix times the vector X,Y,Z,

$$\begin{bmatrix} 1 & -c_1 & 0 \\ 0 & -c_2 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_L - c_1 Y_L \\ Z_L - c_2 Y_L \end{bmatrix}$$

Each point on each image contributes and equation such as this. A three image intersection will therefore yield six equations and three unknowns, requiring least squares for estimation.

insert 3 args: X,Y,Z


```
% coll.m  4-feb-09
% collinearity equation evaluation
function F=coll(x,y,w,p,k,XL,YL,ZL,x0,y0,f)
Mw=[1 0 0;0 cos(w) sin(w);0 -sin(w) cos(w)];
Mp=[cos(p) 0 -sin(p);0 1 0;sin(p) 0 cos(p)];
Mk=[cos(k) sin(k) 0;-sin(k) cos(k) 0;0 0 1];
M=Mk*mp*mw;
UVW=M*[X-XL; Y-YL; Z-ZL];
Fx=x-x0 + f*UVW(1)/UVW(3);
Fy=y-y0 + f*UVW(2)/UVW(3);
F=[Fx;Fy];
```

should read: Mk*Mp*Mw

```
============================================
============================================

% lincoll.m  4-feb-09
% linearization of collinearity equations by numerical partials
function Res=lincoll(x,y,w,p,k,XL,YL,ZL,X,Y,Z,x0,y0,f)
% result matrix is:
% [ Fx dFx/dw dFx/dp dFx/dk dFx/dXL dFx/dYL dFx/dZL dFX/dX dFx/dY dFx/dZ ]
% [ Fy dFy/dw dFy/dp dFy/dk dFy/dXL dFy/dYL dFy/dZL dFy/dX dFy/dY dFy/dZ ]

Res=zeros(2,10);
Pvec0=[w;p;k;XL;YL;ZL;X;Y;Z];
F0=coll(x,y,w,p,k,XL,YL,ZL,X,Y,Z,x0,y0,f);
Res(1:2,1)=F0;
dP=[1.0e-07;1.0d-07;1.0e-07;1.0e-05;1.0e-05;1.0e-05;1.0e-05;1.0e-05;1.0e-05];
for i=1:9
  Pvec1=Pvec0;
  Pvec1(i)=Pvec1(i) + dP(i);
  w1=Pvec1(1);
  p1=Pvec1(2);
  k1=Pvec1(3);
  XL1=Pvec1(4);
  YL1=Pvec1(5);
  ZL1=Pvec1(6);
  X1=Pvec1(7);
  Y1=Pvec1(8);
  Z1=Pvec1(9);
  F1=coll(x,y,w1,p1,k1,XL1,YL1,ZL1,X1,Y1,Z1,x0,y0,f);
  partials=(F1-F0)/dP(i);
  Res(1:2,i+1);
  end
```

insert: = partials

```
% plot_conf_ell.m  28-jan-09
% plot confidence ellipse from 2x2 covariance matrix
% cov = 2x2 covariance matrix
% global 1=use chi2 and z, 0= use F amd t
% r=redundancy
% prob=probability
% x,y = point loc
% factr = enlargement factor for errors

function result=plot_conf_ell(cov,glob,r,prob,x,y,factr)

alpha=1-prob;
ALL_ZEROS=1;
ONLY_X=2;
ONLY_Y=3;
RANK1=4;
RANK2=5;
cs=0;
if((abs(cov(1,1)) < 1.0e-14) & (abs(cov(2,2)) < 1.0e-14))
  cs=ALL_ZEROS;
elseif((abs(cov(1,1)) > 1.0e-14) & (abs(cov(2,2)) < 1.0e-14))
  cs=ONLY_X;
elseif((abs(cov(1,1)) < 1.0e-14) & (abs(cov(2,2)) > 1.0e-14))
  cs=ONLY_Y;
else
  [V,D]=eig(cov);
  % order by magnitude, largest first
  if(D(2,2) > D(1,1))
    temp=D(1,1);
    D(1,1)=D(2,2);
    D(2,2)=temp;
    tempv=V(:,1);
    V(:,1)=V(:,2);
    V(:,2)=tempv;
    end
  if((D(2,2) / D(1,1)) > 1.0e13)
    cs=RANK1;
  else
    cs=RANK2;
    end
  end

% disp('cs = ');
% cs
switch cs
  case ALL_ZEROS
    % plot a point
    plot(x,y,'r.','linewidth',2);
  case ONLY_X
    % plot a horiz line
    sigx=sqrt(cov(1,1));
    if (glob == 1)
      z=norminv(1-alpha/2,0,1);
      scl=z*factr;
    else
      t=tinv(1-alpha/2,r);
      scl=t*factr;
      end
    a=scl*sigx;
    plot([x-a x+a],[y y],'r-','linewidth',2);
  case ONLY_Y
    % plot a vertical line
    sigx=sqrt(cov(2,2));
```

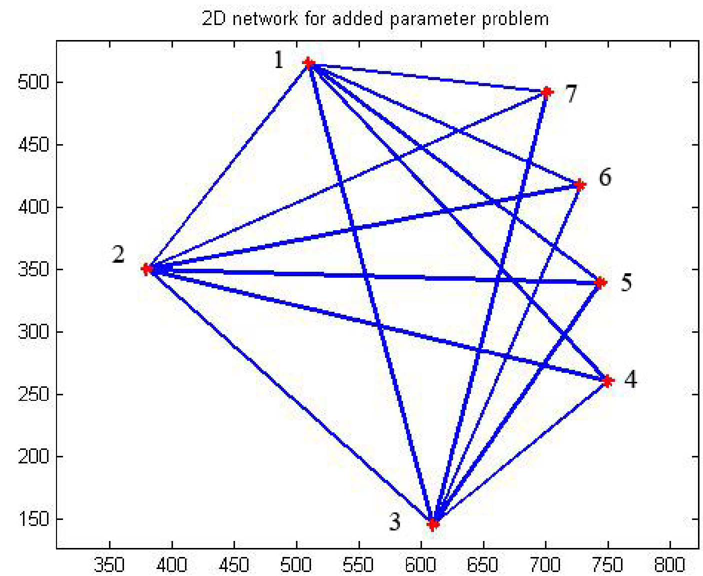
```
  if (glob == 1)
    z=norminv(1-alpha/2,0,1);
    scl=z*factr;
  else
    t=tinv(1-alpha/2,r);
    scl=t*factr;
    end
  a=scl*sigx;
  plot([x x],[y-a y+a],'r-','linewidth',2);
case RANK1
  % plot the tilted line
  vec=V(:,1);
  if(glob == 1)
    a=sqrt(chi2inv(prob,2)*D(1,1));
    a=a*factr;
  else
    a=sqrt(2*finv(prob,2,r)*D(1,1));
    a=a*factr;
    end
  plot([x-vec(1)*a x+vec(1)*a],[y-vec(2)*a y+vec(2)*a],'r-','linewidth',2);
case RANK2
  % ok plot the ellipse
  vec=V(:,1);
  if(glob == 1)
    a=sqrt(chi2inv(prob,2)*D(1,1));
    a=a*factr;
    b=sqrt(chi2inv(prob,2)*D(2,2));
    b=b*factr;
  else
    a=sqrt(finv(prob,2,r)*D(1,1));
    a=a*factr;
    b=sqrt(finv(prob,2,r)*D(2,2));
    b=b*factr;
    end
  theta=atan2(vec(2),vec(1));
  draw_ell(x,y,a,b,theta);
end
```

```
% draw_ell.m  22-oct-08
% function to draw ellipse

function result=draw_ell(xorg,yorg,a,b,theta)

th=theta;
x0=a;
y0=0;
nseg=50;
dalpha=2*pi/nseg;
for i=1:nseg
  alpha=i*dalpha;
  x1=a*cos(alpha);
  y1=b*sin(alpha);
  px0=xorg + cos(th)*x0 - sin(th)*y0;
  py0=yorg + sin(th)*x0 + cos(th)*y0;
  px1=xorg + cos(th)*x1 - sin(th)*y1;
  py1=yorg + sin(th)*x1 + cos(th)*y1;
  plot([px0 px1],[py0 py1],'r-','linewidth',2);
  if(i == 1)
    hold on
    end
  x0=x1;
  y0=y1;
  end
result=0;
```

Problem 2. 2D network with constraints and added parameters



Fix points two points as X1=510.000, Y1=515.000, X3=610.000, Y3=145.000. Constrain points 4,5,6,7 to lie along a circle. Angle observations are given in the following listing (# - at – from – to – D – M – S). The standard deviations of the angle observations are 1 arc minute.

1	1	7	6	17	15	31.0
2	1	6	5	12	39	57.7
3	1	5	4	9	54	20.6
4	1	4	3	28	9	7.8
5	1	3	2	53	22	45.6
6	2	1	7	27	55	11.1
7	2	7	6	12	57	33.2
8	2	6	5	12	34	31.2
9	2	5	4	11	59	37.7
10	2	4	3	28	2	12.6
11	3	2	1	33	8	18.6
12	3	1	7	29	47	11.1
13	3	7	6	8	45	12.8
14	3	6	5	11	16	22.8
15	3	5	4	15	54	25.9

Make global test, show convergence, parameters, residuals, and 50% error ellipse for point 4 and for the circle center point.