CE 503 homework assignment 3 – Oblique Image Rectification

•Produce a rectified digital image of the area covered by the given oblique photograph covering the campus of the University of Illinois and the surrounding towns of Urbana and Champaign. Include area between E393750m and E 400000m, and N4439250 and N4443000 (UTM zone 16) at a ground sample distance (GSD) of 2m. Produce a .tif file and an ESRI .tfw world file for use by ArcView. Compose an image map in ArcView with coordinate grid and legend. Do it in 3 steps:

•Step 1. Compute the 6-parameter transformation parameters to relate the map image coordinates (the pixel coordinates of the scanned map) and the UTM coordinates. The UTM grid line intersections provide convenient points to use for this computation. Examine the residuals from the transformation to confirm that they are consistent with the map scale (1:24000, ½ mm = 12m). Hand in results from this step on Friday, 19 September.

•Step 2. Select points that are visible and identifiable in both the image and the map and are distributed over the image area (this will be a challenge – have patience). Measure the map image coordinates of these and transform via results of step 1 into UTM. Measure the oblique image coordinates of same points. Assuming a plane surface in the object (it is not) compute the 8-parameter transformation between the ground coordinates (UTM) and the image coordinates (pixels). Be sure everything is right handed. Examine the residuals from this transformation (in image space) to confirm good quality result – I am expecting 10-15 pixels, this will depend on the level of redundancy. Hand in Results from this step on Wednesday, 24 September.

•Step 3. Modify the given c-language program “rec8m.c” to match your filenames, file sizes, your transformation parameters, your GSD, your limits, etc., compile under visual studio, and run. You will need to convert the original .tif into a .raw and the output .raw back into a .tif. Photoshop does this. Construct an appropriate ESRI .tfw world file for georeferencing the rectified image. Bring this up in ArcView and confirm that the cursor coordinates are tracking correctly. Make an Image map in ArcView with coordinate reference, legend, scale, and other useful annotations. Make it only for A4 or 8.5x11in. size. Hand in these final results on Wednesday, October 1.

Useful Info:

$$X = a_0 + a_1 x + a_2 y$$
$$Y = b_0 + b_1 x + b_2 y$$

6-parameter transformation

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} a_0 \\ b_0 \end{bmatrix} + \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Form for application

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

Form for computation of the parameters

With the 6-parameter transformation, 3 points will generate 6 equations which will yield a unique solution (dangerous !). More points (redundancy) will require making a least squares adjustment of the data.

$$\begin{bmatrix} X_1 \\ Y_1 \\ X_2 \\ Y_2 \\ \vdots \\ X_n \\ Y_n \end{bmatrix} \approx \begin{bmatrix} 1 & x_1 & y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_1 & y_1 \\ 1 & x_2 & y_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & y_n & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_n & y_n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

This is a matrix equation, with LS solution,

$$\mathbf{b} = \mathbf{Ax}, \quad \text{or}$$

$$\mathbf{Ax} = \mathbf{b},$$

solve overdetermined system

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

residuals given by

$$\mathbf{v} = \mathbf{b} - \mathbf{Ax}$$

Note: Matlab is a great environment for doing the matrix calculations shown here.

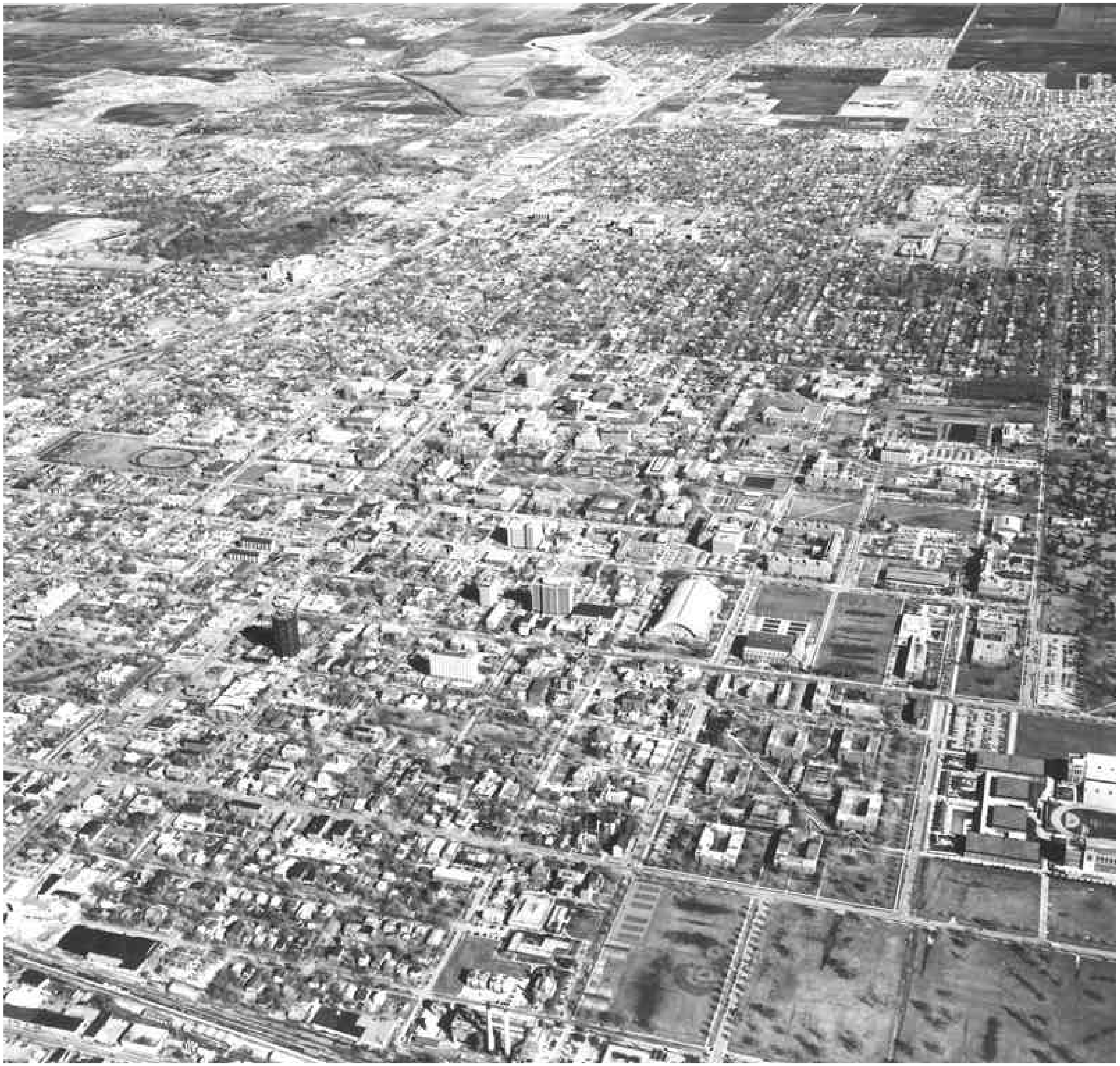
Use Notepad or the Matlab editor to create a .m file of commands, then from Matlab command line, call the file to execute it. Record results with “diary filename” and “diary off” commands.

Data can be found on read-only share:

\\geomatics\data\bethel\ce503\urbana

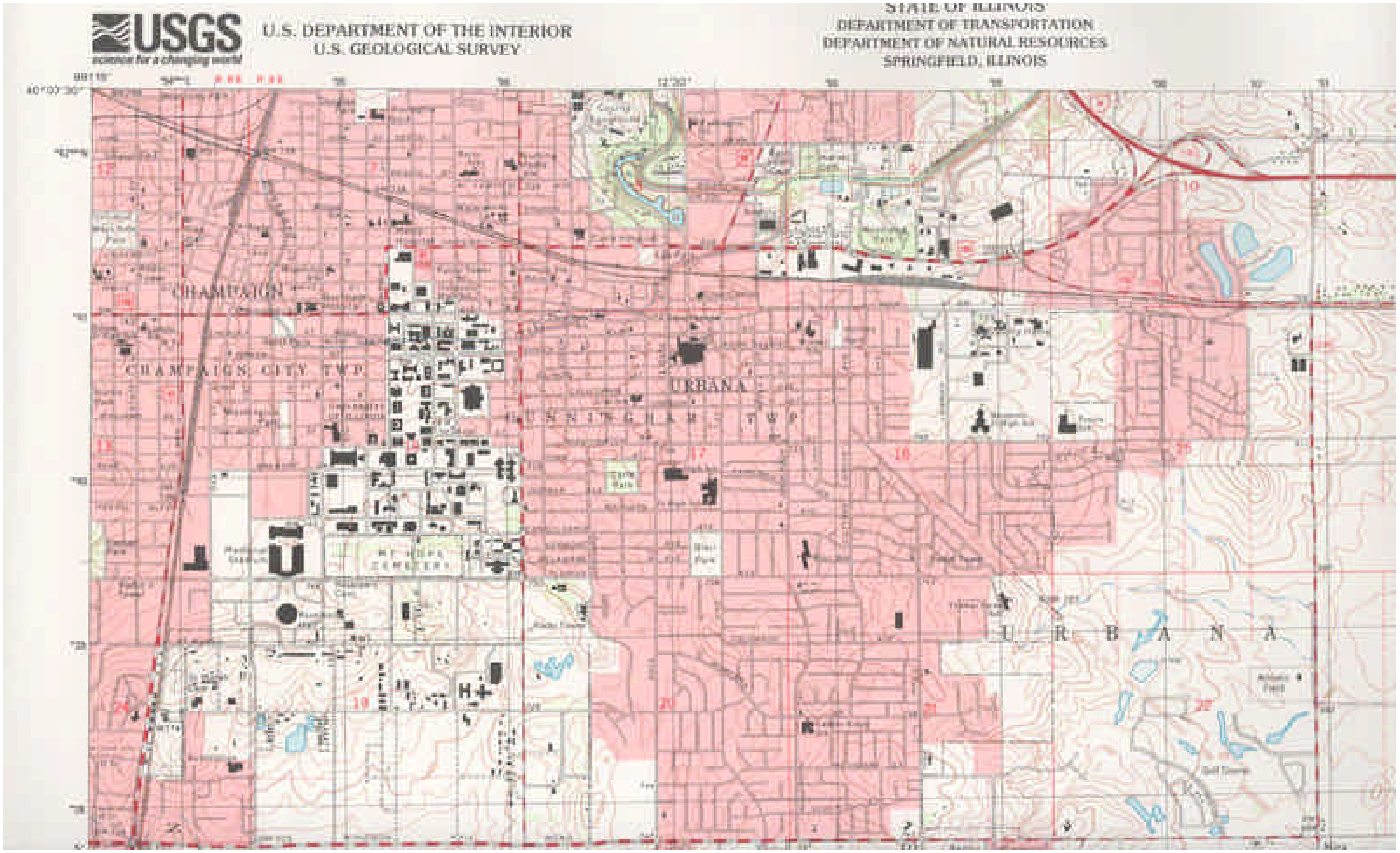
Small browse .jpg files are in the main directory (uiuc_sm.jpg, urbana.jpg) the large files to make the actual measurements are in subdirectory “big” (uiuc.tif, urbana.tif)

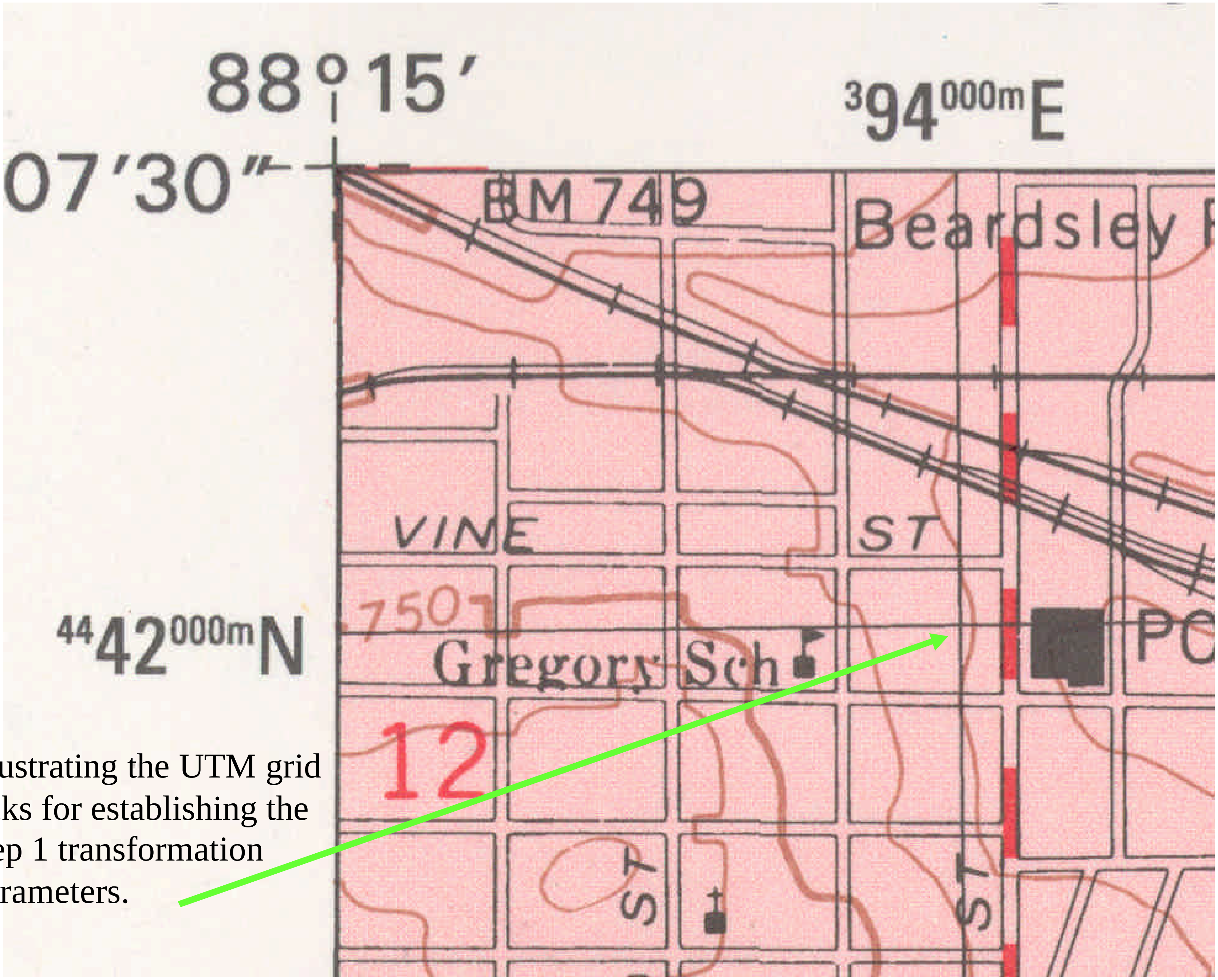
I will have some CD’s outside 4109 in case ECN is slow to set up the share access.



Oblique view looking ~East over the project area. Note large scale in the foreground and small scale in the background. Rectification should produce a uniform scale on output but will the spatial *resolution* be uniform? What about sampling and aliasing effects? And what happens to relief displacment with simple rectification?

USGS 7.5 minute quadrangle map (1:24000) scanned at 600 dpi. Scanning is *approximately* aligned with east-north axes but not exactly – that is what the step 1 6-parameter transformation will accomplish. Make your actual measurements from the big (175MB) .tif file.





Illustrating the UTM grid ticks for establishing the step 1 transformation parameters.

Eight Parameter Transformation - Ground XY (plane) to image

$$x = \frac{a_0 + a_1 X + a_2 Y}{1 + c_1 X + c_2 Y}$$

$$y = \frac{b_0 + b_1 X + b_2 Y}{1 + c_1 X + c_2 Y}$$

Equations for application of the transformation

$$x + c_1 xX + c_2 xY = a_0 + a_1 X + a_2 Y$$
$$y + c_1 yX + c_2 yY = b_0 + b_1 X + b_2 Y$$

Rearrange, multiply by denominator

$$x = a_0 + a_1 X + a_2 Y - c_1 xX - c_2 xY$$
$$y = b_0 + b_1 X + b_2 Y - c_1 yX - c_2 yY$$

make into pseudo-linear equations for ease of computation

Expressed in Matrix-Vector Form

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & X & Y & 0 & 0 & 0 & -xX & -xY \\ 0 & 0 & 0 & 1 & X & Y & -yX & -yY \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ b_0 \\ b_1 \\ b_2 \\ c_1 \\ c_2 \end{bmatrix}$$

Matrix-vector equation for 1 point

$$\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \\ x_n \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & X_1 & Y_1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 \\ 0 & 0 & 0 & 1 & X_1 & Y_1 & -y_1X_1 & -y_1Y_1 \\ 1 & X_2 & Y_2 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 \\ 0 & 0 & 0 & 1 & X_2 & Y_2 & -y_2X_2 & -y_2Y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_n & Y_n & 0 & 0 & 0 & -x_nX_n & -x_nY_n \\ 0 & 0 & 0 & 1 & X_n & Y_n & -y_nX_n & -y_nY_n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ b_0 \\ b_1 \\ b_2 \\ c_1 \\ c_2 \end{bmatrix}$$

Matrix-vector equation for n points

The ground to image transformation should be setup and solved in the order

(x,y) <= (X,Y),   X,Y: Ground, x,y: image

As with prior transformation, the matrix form of the equations and the LS solution can be expressed as,

$$\mathbf{Ax} = \mathbf{b}$$

$$\mathbf{x} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{v} = \mathbf{b} - \mathbf{Ax}$$

For CE506 students, note that the residuals here are *minus* the ones we derive in Adjustment class. This is the "linear algebra" notation & convention. We must be multi-lingual. They both work as long as your are consistent within a problem and apply the corrections with the proper sign!

```
/* rec8m.c 16-sep-03 */
/* rectify an oblique image using 8-parameter */
/* transformation */
/* modify the numbers in RED and compile, and run */
/* note the supplied .tif file must be converted to .raw */
/* the output .raw must be converted back to .tif */
/*
note the corresponding .tfw world file for arcview should
be an ascii text file (create with notepad) with entries
+GSD
0
0
-GSD
UPPER LEFT X
UPPER LEFT Y
*/

/*
compile and execute under unix by

  cc -o rec8m rec8m.c -lm
  rec8m

compile under windows by

  nmake -f rec8m.mak
  rec8m
  (you may need to run vcvars32.bat first, alternatively
  compile in visual studio)

*/

/*
variable defn for bilinear interp

       i0,j0                   i0,j1
       *-----------------------*
       |(1)              |  (2)|
       |                 |     |
       |                 |     |
       |                 |     |
       |                 |     |fracr
       |                 |     |
       |                 |     |
       |                 |     |
       |                f|     |
     t0*-----------------*-----*t1
       |                 |     |
       |                 |     |
       |(4)              |  (3)|
       *-----------------------*
       i1,j0     fracc         i1,j1

*/


#include <stdlib.h>
```

```
#include <stdio.h>
#include <math.h>

/* setup for purdue engineering mall monochrome image */

#define INROWS  750
#define INCOLS 1035
#define GSD 0.1
#define UPLEFTX 14326.0
#define UPLEFTY 75208.0
#define OUTROWS 1610
#define OUTCOLS 1110

/* transformation from ground to photo */

#define A0 -50.95961490909847
#define A1   1.28220453648728
#define A2  -4.65200062086704
#define B0   6.20993099113558
#define B1  11.27991314142943
#define B2   2.89543630302745
#define C1  -0.00072239030414
#define C2   0.00449064809130

#define RMEAN 3.126250000000000e+02
#define CMEAN 5.092500000000000e+02
#define XMEAN 1.438875000000000e+04
#define YMEAN 7.512375000000000e+04

/* select one of these by =1 */
#define NEAREST_NEIGHBOR 1
#define BILINEAR 0

/* allocate arrays for monochrome images */
unsigned char inpic[INROWS][INCOLS];
unsigned char outpic[OUTROWS][OUTCOLS];

int main(int argc, char *argv[])
{
FILE *inread,*outwrite;
int i,j;
int i0,i1,j0,j1;
double pr,pc,prm,pcm;
int ipr,ipc;
double runx,runy,numx,numy,den;
double fracr,fracc;
unsigned char gra;
double m1,m2,m3,m4;
double t0,t1;
int test;

/* open files - put full pathname if needed */
if ((inread=fopen("emall.raw", "rb")) == NULL)
  {
  printf ("error opening emall.raw");
  exit (1) ;
  }
```

```
if ((outwrite=fopen("emallrec.raw", "wb")) == NULL)
  {
  printf ("error opening emallrec.raw");
  exit (1) ;
  }

printf("read data\n");
for(i=0; i<INROWS; i++)
  {
  fread(inpic[i],INCOLS,1,inread);
  }
fclose(inread);

printf("resample the image\n");
for(i=0; i<OUTROWS; i++)
  {
  test= i % 100;
  if(test == 0)
    {
    printf("row number %d\n",i);
    }

  runy=(UPLEFTY-YMEAN) - i*GSD;
  for(j=0; j<OUTCOLS; j++)
    {
    runx=(UPLEFTX-XMEAN) + j*GSD;
    numx=A0 + A1*runx + A2*runy;
    numy=B0 + B1*runx + B2*runy;
    den=1.0 + C1*runx + C2*runy;

    pr=numx/den;
    pc=numy/den;
    /* add the mean back into the row & column */
    prm=pr + RMEAN;
    pcm=pc + CMEAN;

    gra=128;
    if(NEAREST_NEIGHBOR == 1)
      {
      ipr= (int) (prm+0.5) ;
      ipc= (int) (pcm+0.5);
      if((ipr >= 0) && (ipr < INROWS) &&
       (ipc >= 0) && (ipc < INCOLS))
      {
        gra=inpic[ipr][ipc];
      }
      }

    /* bilinear */
    if(BILINEAR == 1)
      {
      i0=(int) prm;
      fracr=prm - (double) i0;
      i1=i0+1;
      j0=(int) pcm;
      fracc=pcm - (double) j0;
      j1=j0+1;
      if((i0 >= 0) && (i1 < INROWS) &&
       (j0 >= 0) && (j1 < INCOLS))
      {
      /* ------------------------- */
      m1=(double) inpic[i0][j0];
      m2=(double) inpic[i0][j1];
      m3=(double) inpic[i1][j1];
      m4=(double) inpic[i1][j0];
      t0=fracr*m4 + (1-fracr)*m1;
      t1=fracr*m3 + (1-fracr)*m2;
      gra=fracc*t1 + (1-fracc)*t0;
      }
      }
    outpic[i][j]=gra;
    }
  }

printf("write data\n");
for(i=0; i<OUTROWS; i++)
  {
  fwrite(outpic[i],OUTCOLS,1,outwrite);
  }
fclose(outwrite);

}
```