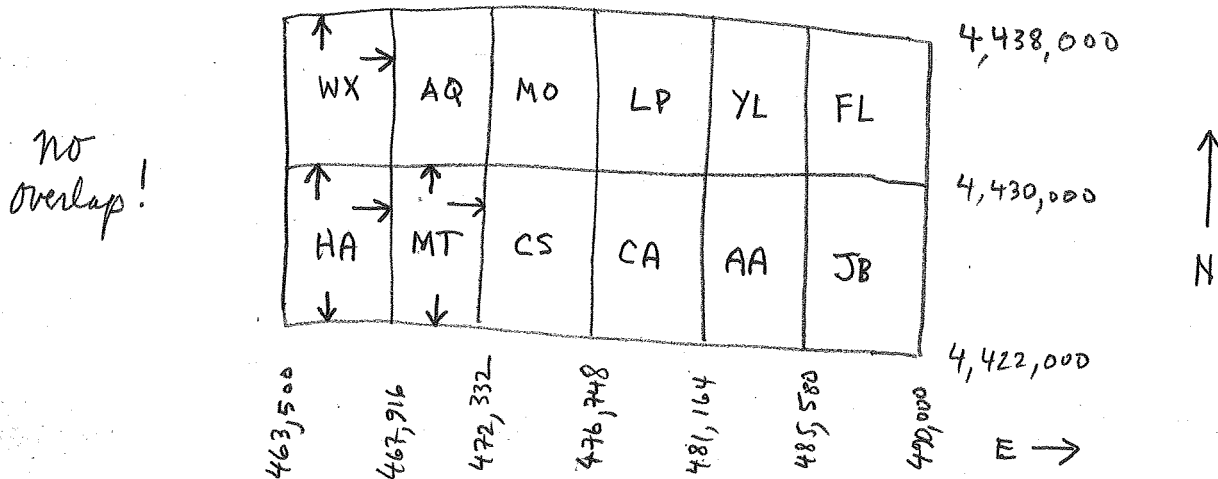


# Sat. Photogrammetry HW5

Orthorectification  $\neq$  vector overlay

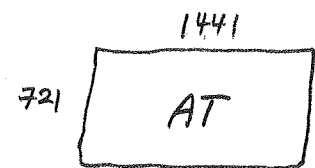
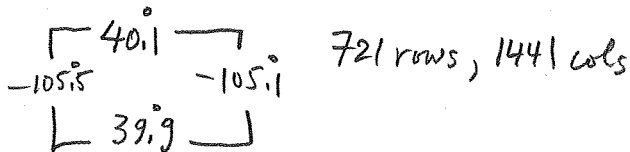
1/7

Using results for RPC's for image #1 from HW4, orthorectify your patch of the given extent [UTM Z13, meters]



GSD of output should be 2m, interpolate from downsampled image.

Elevations @ 1 arc second intervals in bc.dem row-wise from N → S



```

Read by      fid = fopen(filename, 'r');
             A = fread(fid, [1441, 721], 'single');
             AT = A';
    
```

Lower Left Corner lat (dec. deg.) 39.9

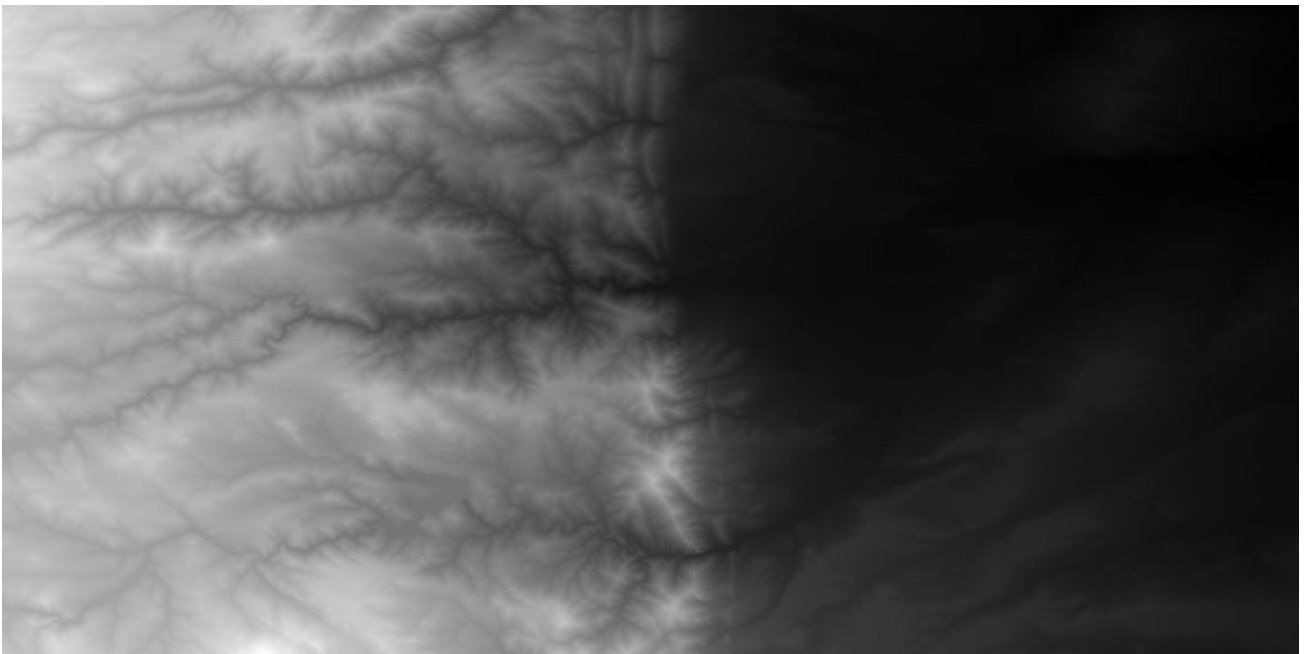
Lower Left Corner long (dec. deg.) -105.5

DEM: interval (dec. deg.) 0.000277777777777778 (1 arc second!)

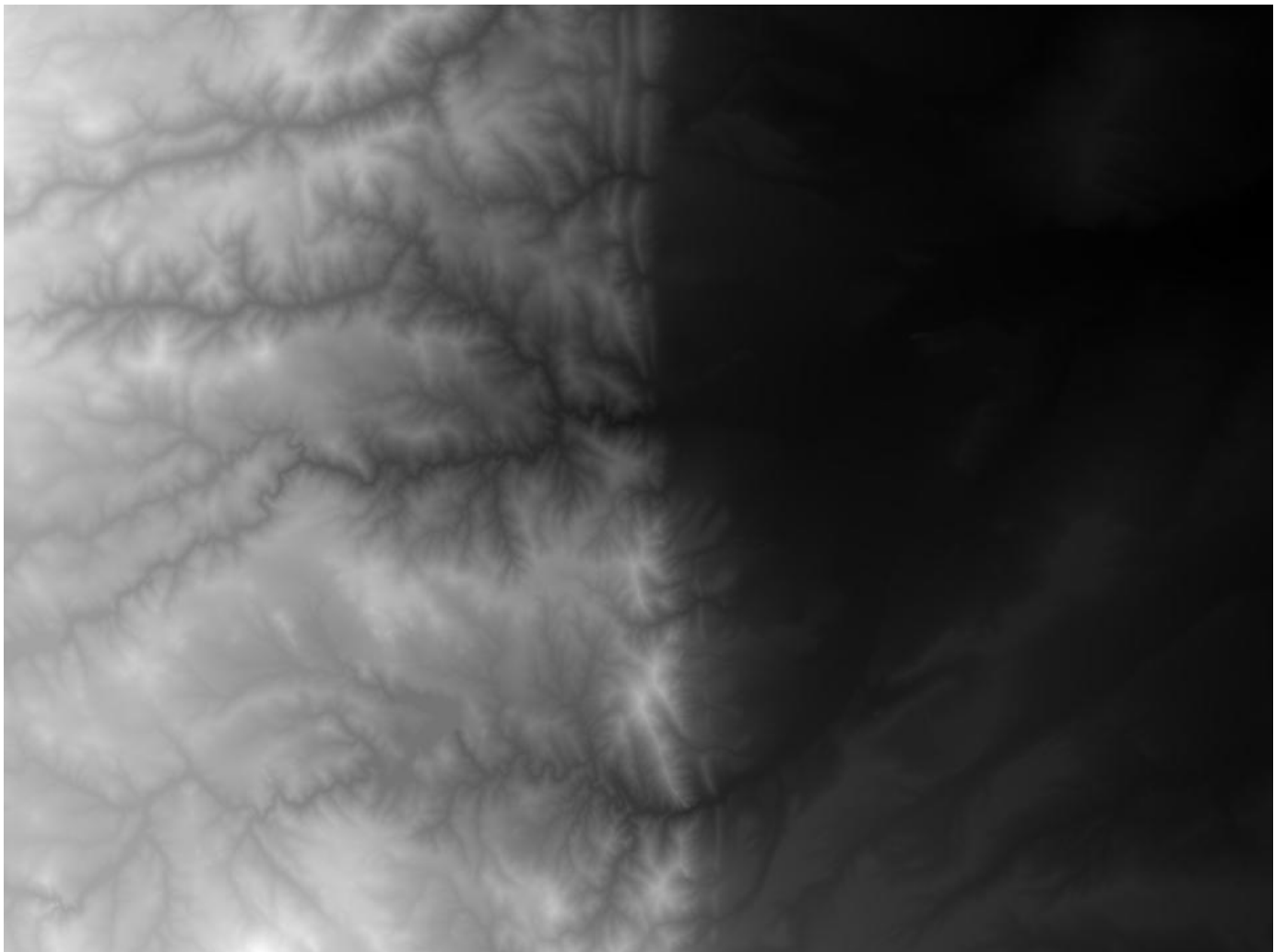
num. rows 721

num. cols 1441

Data in DEM file is orthometric height,  $H$ , in meters



BC.DEM 1 arc second DEM shown as gray scale (white = high, black = low) with equal spacing, so aspect ratio is wrong



BC.DEM shown with correct spacing so the aspect ratio is correct

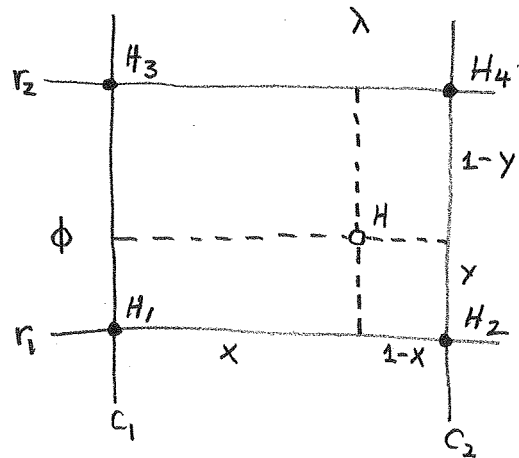
lets reverse order so rows are stored  $S \rightarrow N$

2/7

```
B = zeros(721, 1441, 'single');
for i = 1:721
    B(i, :) = AT(722-i, :)
end
clear A AT
```

Now we need to do index arithmetic to get rows and columns,

```
dphi = phi - phi_LL;
dsec = dphi / interval;
r1 = fix(dsec) + 1;
r2 = r1 + 1;
gamma = dsec - fix(dsec);
dlambda = lambda - lambda_LL;
dsec = dlambda / interval;
c1 = fix(dsec) + 1;
c2 = c1 + 1;
x = dsec - fix(dsec);
```

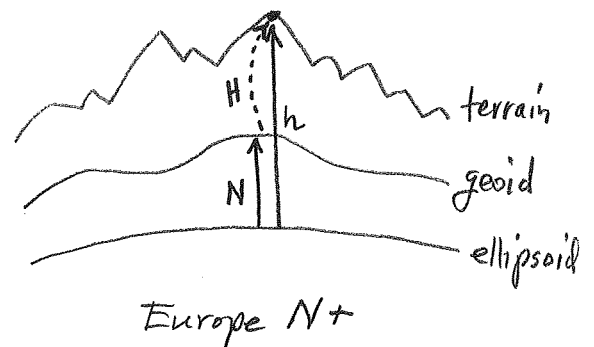
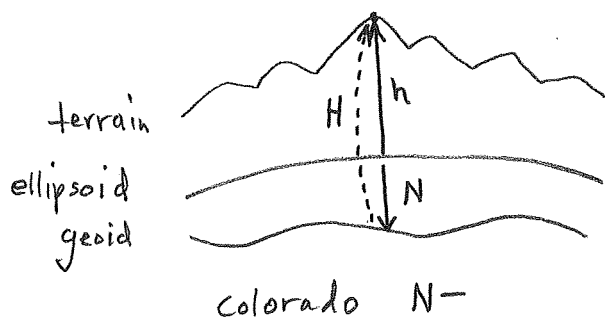


Bilinear interpolation for height at  $(\phi, \lambda)$

```
H1 = B(r1, c1);
H2 = B(r1, c2);
H3 = B(r2, c1);
H4 = B(r2, c2);
```

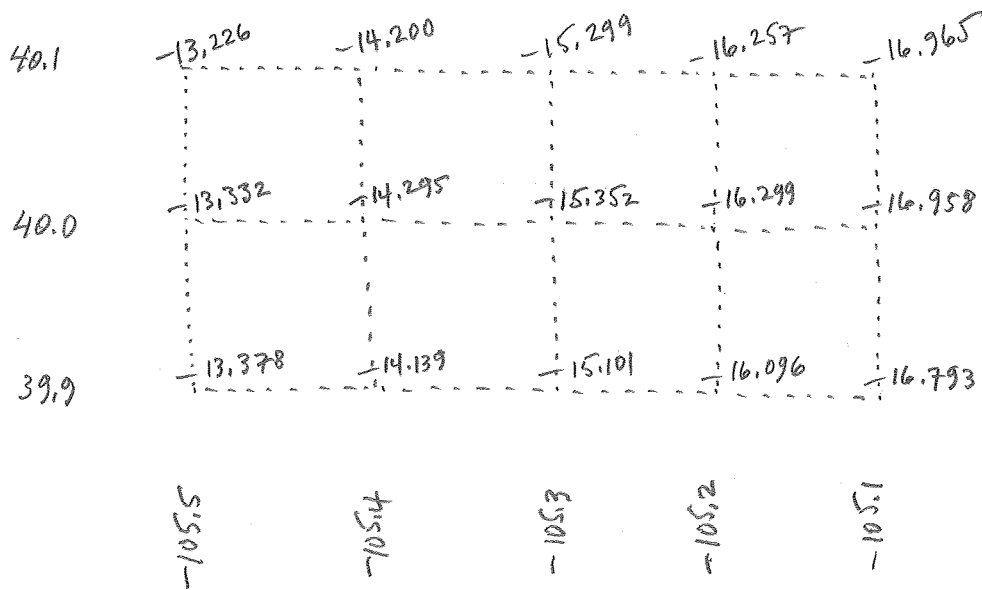
$$H = (1-x-y+xy)H_1 + (x-xy)H_2 + (y-xy)H_3 + (xy)H_4$$

Convert to ellipsoid height,  $h = H + N$



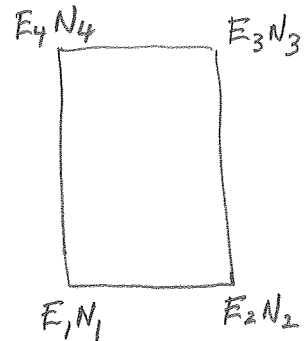
Interpolate a value for N at each point, using the grid:

3/7



These numbers come from NGS Geoid12a interactive calculator

Create a clipping rectangle for your area:  
(in matlab)



Bdry.Geometry = 'Polygon';

Bdry.X = [E1 E2 E3 E4 E1 NaN];

Bdry.Y = [N1 N2 N3 N4 N1 NaN];

Bdry.Name = 'BoulderOrtho';

→ Bdry.BoundingBox = [min(Bdry.X) min(Bdry.Y) max(Bdry.X) max(Bdry.Y)];  
Shapewrite(Bdry, 'boundary');



This will write files  
boundary.shp  
boundary.shx  
boundary.dbf

You will use this to clip the county road data to just overlay your image sections.

To clip the road layers :

4/7

Start ArcMap

- blank map
- connect to folder  (catalog tab)
- add data 
  - Highways
  - Major Roads
  - Local Roads
  - Rail Lines 100K
  - boundary (from before)
- geoprocessing
  - clip
    - clip dialogue    input feature
    - clip feature
    - output feature
    - XY tolerance 0.5
  - OK

---

Since we have  $GSD = 2\text{ m}$  in ortho image, we must interpolate in downsampled (Low Pass Filtered) image to avoid aliasing.

Find `col01-4.jpg` in the original image archive.

In the rectification code, obtain  $l, s$  for current point, then

$$l_p = l/4 \quad \text{and use } l_p, s_p \text{ to interpolate in}$$
$$s_p = s/4$$

the downsampled image.

# Template/Flowchart for orthorectification code

5/7

```

in_img = imread('col1-4.jpg', 'JPEG');
[nrows, ncols] = size(in_img);
out_img = zeros(nrows, ncols, 3, 'uint8');
for i = 1:nrows

```

```

    for j = 1:ncols
        E = Emin + (j-1)*2;
        N = Nmax - (i-1)*2;

```

transform to  $\phi, \lambda$

interpolate H from DEM

Interpolate N from Geoid grid

compute h

$$\begin{bmatrix} l \\ s \end{bmatrix} = \text{RPC\_G2I}(\phi, \lambda, h);$$

$$lp = l/4$$

$$sp = s/4$$

if  $lp, sp$  outside image ( $< 1, > \text{max}$ )

```

    r = 128;
    g = 128;
    b = 128;

```

else

bilinear interpolation for intensity from in\_img

```

    r = intensity;
    g = intensity;
    b = intensity;
end

```

```

out_img(i,j,1) = r;
out_img(i,j,2) = g;
out_img(i,j,3) = b;
end

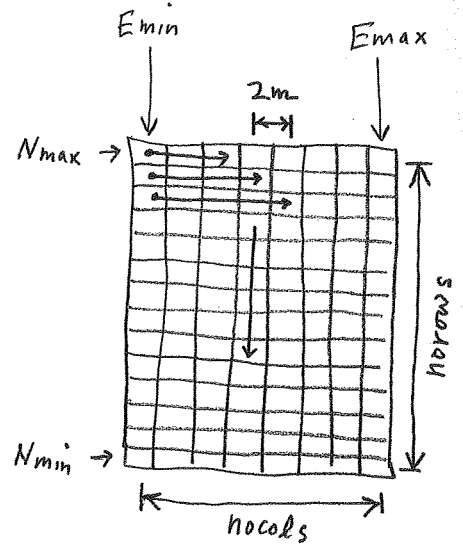
```

end

```

imwrite(out_img, 'outfile.jpg', 'JPEG');

```



suggest reading  
RPC parameters  
offsets, scales,  
A, B, C, D into global  
variables for use  
by function

You will need:

ftmgeo\_utm z13.m

$$\begin{bmatrix} \phi \\ \lambda \end{bmatrix} = \text{ftmgeo\_utm z13}(E, N);$$

← radians (!)                      ↑ meters

Suggest for Debug that you start with either larger GSD or smaller extent so the program runs quickly, then when it seems OK, run with real values (it will take a long time). Otherwise, during debug, you just waste time waiting.

Construct, by hand, the necessary "ESRI World File" to go with your output .jpg file:

□ .jpg

□ .jgw

← text file, create with notepad, etc. 6 numbers

GSD X  
0  
0  
-GSD Y  
X upper left  
Y upper left



2  
0  
0  
-2  
E min  
N max

This will let ArcGIS place the image in correct location.

Hand in:

1. Code: hardcopy + email (+ functions)
2. Maps: Image + vector overlay, arrange intensities and colors so things are visible. Hardcopy.
3. Zoom in to show good registration. Hardcopy.
4. Zoom in to show any areas with poor registration. Why is it poor? Hardcopy.
5. □.jpg, □.jgw

I will merge all of the files into a (hopefully) seamless MOSAIC!

This will be graded like Exam # 2.

get photo2-15-hw5-files.zip from  
[ftp://ftp.ecn.purdue.edu/bethel/](http://ftp.ecn.purdue.edu/bethel/)

that contains bc.dem  
ftmgeo-atm213.m  
+ all shape files } ~ 7MB

When you take intensity values from image you must convert to double before doing math, likewise when done doing math you must convert back to uint8 before storing in image array.

```
gd = double(g)
gu = uint8(round(gd))
gd must be between 0 & 255
```