Floodplain Redesign Project

Christian Braudrick, UCB
Glen Leverich, SFSU
With invaluable technical assistance from Joel Rowland, UCB

Photogrammetry Technique Manual for the Small and Big Basins

(last revised 2/10/06)

The purpose of transforming images taken by the camera angled toward the basin is to deal with the perspective and distortion of the image due to its specific angle. The angle is important because it sees the bends in the laser line that are created by topographic changes on the floodplain. The exact pixel sizes can be determined using Photoshop and Excel. The image is then transformed in Matlab to create a single cross-section or complete 3D contour map that reflects near exact elevations.

To Calibrate Basin Surveys:

<u>Photograph a grid on the basin</u>

First step is to photograph a grid with either 10x10cm or 20x20cm squares onto the basin bed at the lowest elevation; i.e.: below the sand and furthest downslope along the basin. The grid needs to be supported in a near perfect vertical position and perpendicular to the basin's long axis (see Figure 1). Place the grid in plane with the laser that has already been mounted on the carriage. The laser and camera should have already been mounted in their appropriate positions and locked down to prevent refocusing or zoom changes (set focal length). Take a photograph of the grid once everything is in place. Physically measure the distance between the camera lens and the image center point on the grid (look at the image to determine where the centerpoint of the image appears on the grid; Figure2).

Figure 1. Grid Set-up for 3-Camera setup in the Big Basin.

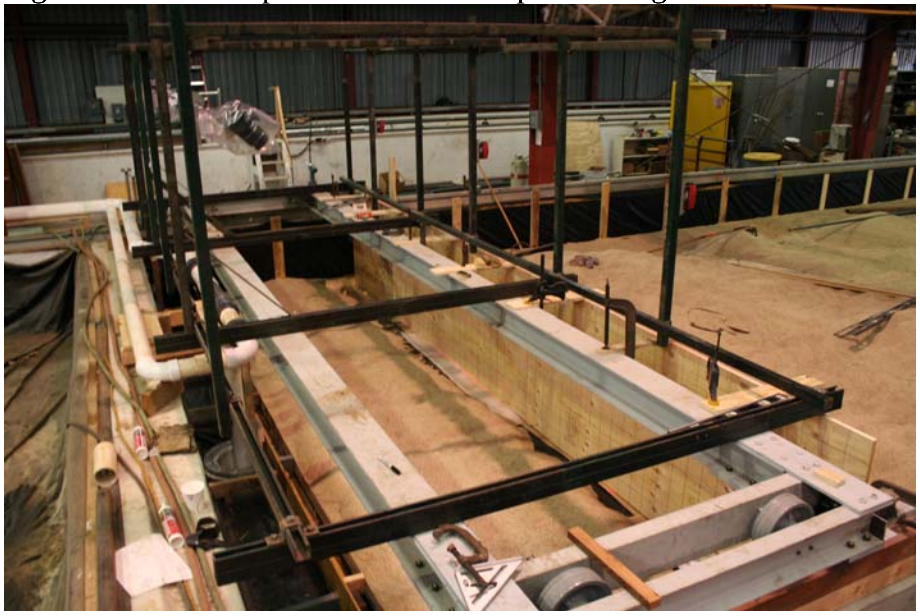
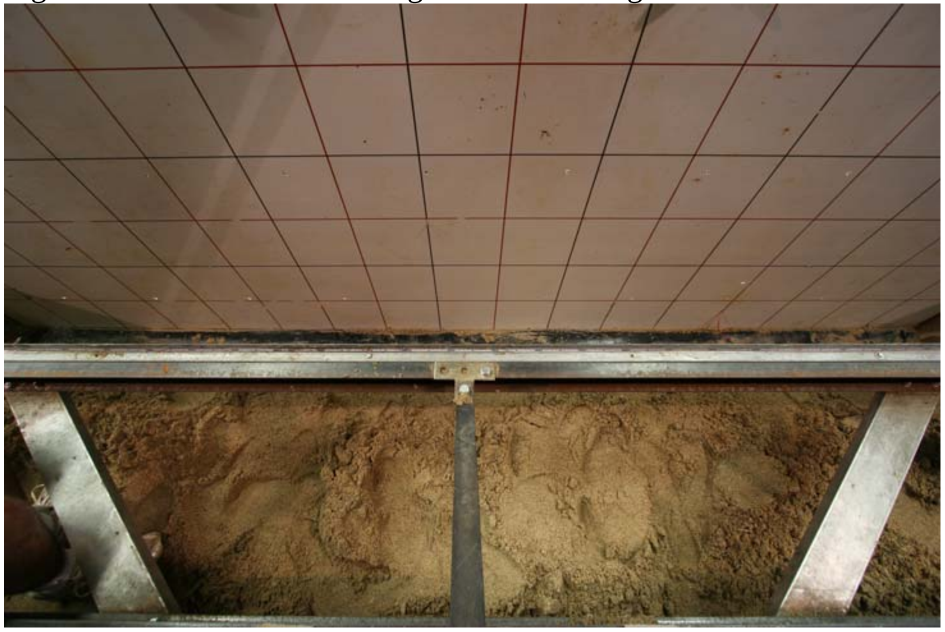


Figure 2. Small Basin Grid Image taken at 60-degrees from horizontal.



Photoshop Lens Distortion Correction with PTLens

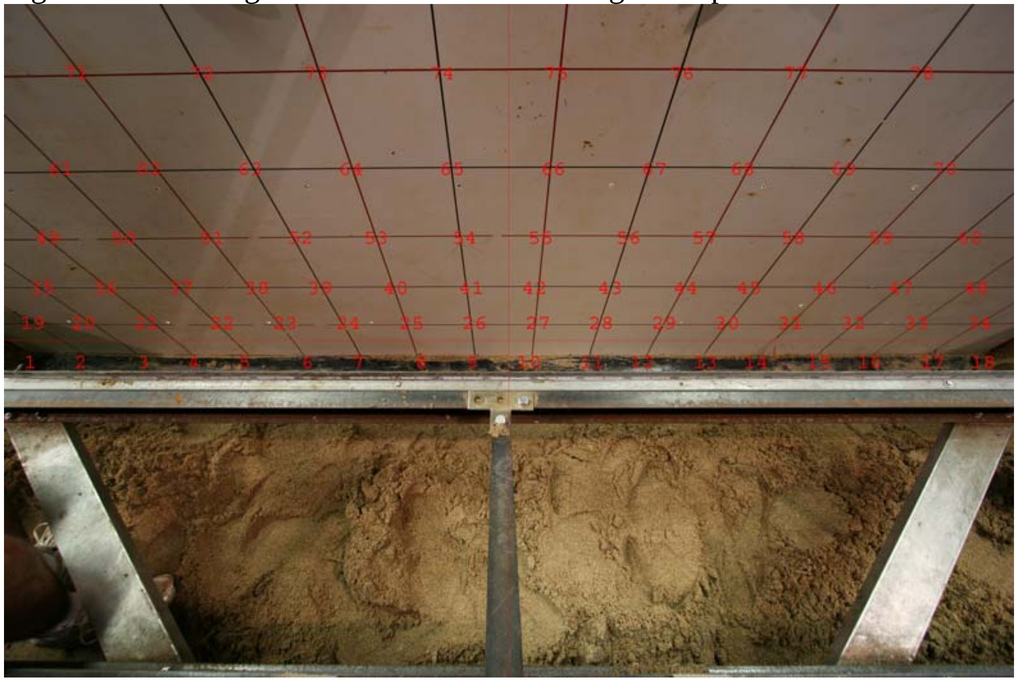
Correcting the image distortion caused by the short focal length of the camera may be necessary prior to determining the node locations. Download the PTLens program from http://epaperpress.com/ptlens/ and follow the installation instructions carefully. Test an image with the program to see if it is working properly before batch processing a series of survey images. To batch process a series of survey images create an action first to automate the batch process. First open a test image. Open the actions palette and press the "Create New Action" button (located at the bottom of the actions palette). Name the action; e.g.: "PTLens Correction". Click "Record" and Photoshop begins recording your actions automatically. Perform the following steps on the open image: click open the "Filter" Menu→ "ePaperPress"→ "PTLens". The PTLens window will open. Make sure the correct camera information and focal length set when the image was taken is entered into the fields. Click on "options" and unclick "always display dialog" under preferences. Keep the other preferences selected. Click "Okay". Now save this corrected image with a new name and in the same folder as the original image. Click the "stop" button in the Actions palette.

Test some images to batch process. First set up two new folders somewhere in your directory and place a group of test images in the first folder. Name the folders something like "test" and "test ptlens corr". In Photoshop, click "File"→ "automate" → and "Batch…". The Batch window is now open. Under Play and Action, select the "PTLens Correction" action you just created. Next to "Source:", click the down arrow tab and select "folder". Click "Choose" and browse for the first folder that contains the uncorrected images. Make sure there are no other files or images that should not be corrected because Photoshop will batch process every file or image in this folder. Next to "Destination", click the down arrow tab and choose "Folder". Click "Choose" and browse for the second folder that is empty. Photoshop will save the corrected images into this folder during the batch process. Check the box next to "Override

Action “Save As” Commands”. This changes how Photoshop will save the corrected images. Under “File Naming” select how you want the corrected images to be named. Do the following in order to later process these images in Matlab: type “Image” in the first box, select “4 Digit Serial Number” in the second box, and select “extension” in the third box. Click “OK”. Photoshop will batch process the images in the first folder, correct the distortion with PTLens, and will rename and save the corrected images in the second folder.

Determine Node Locations in Photoshop and Excel

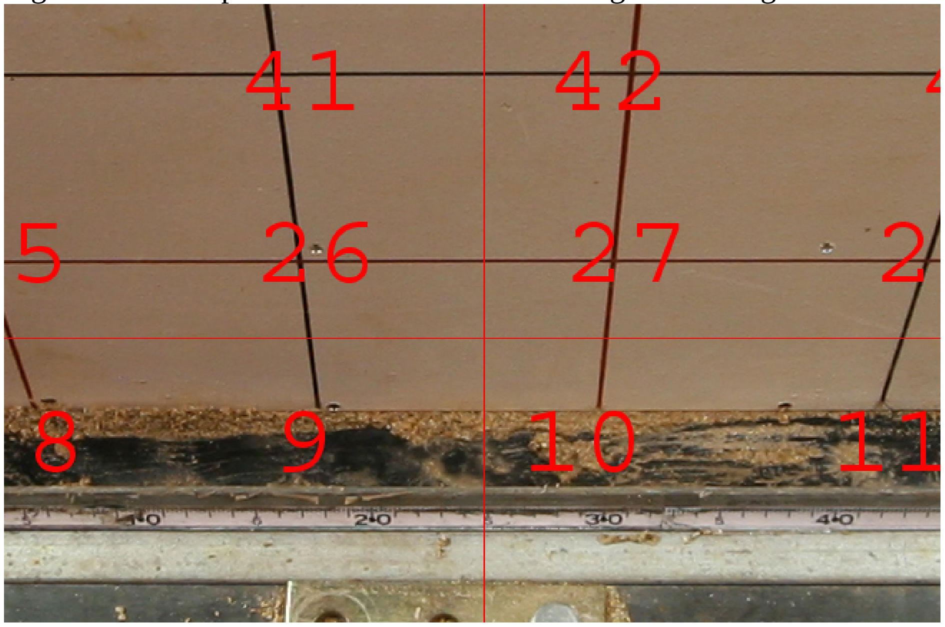
Once the grid calibration image has been taken, use Photoshop or some other image viewing program that allows you to locate pixel locations. The nodes are the intercepts of the horizontal and vertical lines on the grid. Keep track of the nodes by labeling them in Photoshop; e.g.: number them 1, 2, 3, etc. In Excel make a column for the node number. Also, make two columns for the X and Y positions of the nodes in pixel values. In Photoshop, record each node location in X and Y coordinates; e.g.: node #2 = 118, 1307, where x=118 and y=1307. These values need to be in pixels (see Figure 3).

Figure 3. Grid Image from Small Basin showing centerpoint and numbered nodes.


Next step is to determine the width and height that each pixel needs to be in order to transform the image. The best way to do this is by looking at the grid square that the centerpoint falls into (see Figure 3). This square has the least amount of distortion and therefore is the best representation of the width and height of each transformed pixel. To find the width, first draw a centerline that bisects the image vertically, which should go through the image centerpoint. For example, if your image is ~8MP (3456x2304pixels), the bisecting line is at y=2304/2=1152pixels. Measure the length of the line between its intercepts with the center square (small basin=207pixels and big basin=111pixels across the center square). The transformed pixel width is the square size divided by the measure pixels. For example, the small basin pixels = 200mm/207pixels = 0.9662mm wide.

Determining the vertical size of the transformed pixels is a little trickier because the image is not in the same plane as the grid. You can see this in the image because the squares above the center become taller, while the squares below the center become smaller (see Figures 3 and 4). However, the vertical size of the transformed pixels can also be determined using the center square. Draw a bisecting line on the image that divides the image in half horizontally and passes through the centerpoint. Measure the height of the center square along the bisecting line between the intercepts with the top and bottom of the square. The transformed pixel height is the actual height of the grid square divided by the measured pixels. For example, the small basin transformed pixel height is 200mm/104pixels=1.923mm. We will use this later in Matlab.

Figure 4. Centerpoint of Small Basin Grid Image with image bisector lines and node numbers.



In Excel, create two new columns titled “Trans X” and “Trans Y”. The values for these columns are based on the work you did above in figuring out the transformed pixel dimensions. Begin with the nodes that make up the center square. The location of the centerpoint determines how the center square will be transformed and this is variable for each experimental basin. See the example Excel data tables for the Small Basin and the Big Basin. Plot the data for X vs. Y and Trans X vs. Trans Y on the same graph to see how the nodes have been transformed (use this to compare with the Matlab work later).

Copy only the X and Y (non-transformed) column data into a new Excel file. Do not copy any headings or notes; just the actual cells containing X and Y pixel data. Save as “XY.txt” (Text (Tab delimited)). Do the same for Trans X and Trans Y column data and save as “XYtrans.txt). Important: delete any rows with data points that are beyond the image dimensions (3456:2304). For example, delete rows in the TransX and Y columns and corresponding rows in the XY data columns if values for X<1 or >3456, and/or Y<1 or >2304. You should end up with the same number of columns and rows for both XY.txt and XYtrans.txt.

Matlab Image Processing using Matlab 7

The first step is to create an image transformation matrix that will be used during the transformation of all images for the specific basin. The transformation function will remove image distortion and is performed using a predetermined “Tform” matrix that is specific to a given camera/survey setup.

Start Matlab. Browse your folders in the Current Directory window to find the XY.txt and XYtrans.txt files. This window is located in the top left of your screen above “Command History” and left of “Command Window” (click the “Current Directory” tab if the “Workspace” window is shown instead). Select the folder where the XY.txt and XYtrans.txt files are saved. In the Command Window, load in the XY.txt and XYtrans.txt files into Matlab. Type at the prompt:

```
>>load(‘XY.txt’);            <enter>
>>load(‘XYtrans.txt’);       <enter>
```

Note: “>>” never needs to be typed, “;” needs to be entered at the end of the command to prevent Matlab from displaying the created matrix in the Command Window, and “<enter>” means hit enter on your keyboard.

Click the “Workspace” tab to see the two files you just loaded. Matlab has registered them as a matrix file. For example, for the small basin: XY is a 77x2 double matrix file with 77 rows and 2 columns.

Assign values to X, Y, Xtran, and Ytran.

> Note: the following command to be used: >>X=XY(1:77,1);
> This means:
> X is the chosen name of the new matrix you are creating (it is an arbitrary choice)
> XY refers to the XY matrix already created when you loaded XY.txt into Matlab
> The parentheses (…) tell Matlab that you want X to equal some part of XY.
> The left side of the comma refers to row numbers and the right side refers to columns; e.g.: rows 1 through 77 and all of column 1 are being selected above.
> The semi-colon goes at the end of the command and tells Matlab not to display the matrix in the Command Window (try it once and you’ll see why you need it most of the time).

In the “Command Window” prompt, type the following. Place your number of rows where appropriate and hit enter after each line.

```
>>X=XY(1:”number of rows in your x-column”,1);            <enter>
>>Y=XY(1:”number of rows in your y-column”,2);            <enter>
>>Xtran=XYtrans(1:”number of rows in your xtran-column”,1);  <enter>
>>Ytran=XYtrans(1:”number of rows in your ytran-column”,2);  <enter>
```

To create the Transformation function that will be applied to future images, type the following (leave a space between X and Y and Xtran and Ytran, but no where else):

```
>>Tform=cp2tform([X Y],[Xtran Ytran],’projective’);   <enter>
```

This utilized the projective transformation type. There are other transformation types available to use, such as ‘polynomial’, however ‘projective’ is used by the small and big basins and in Joel’s experiment (see Figures 5 and 6). Polynomial may work provided the density of nodes is sufficient (see Figure 6 for example with a low density of nodes).

Figure 5. Preferred Image of Small Basin Grid with Projective Transformation

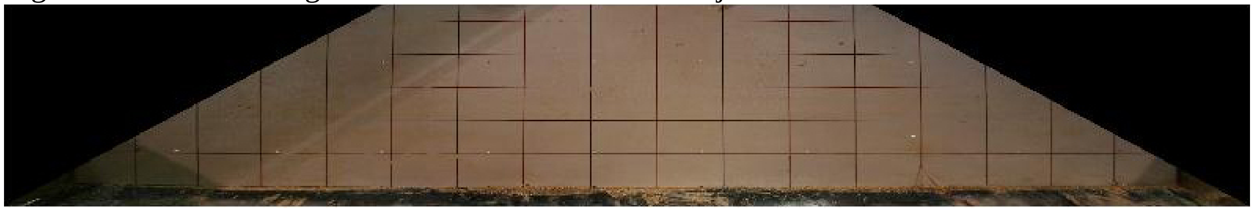
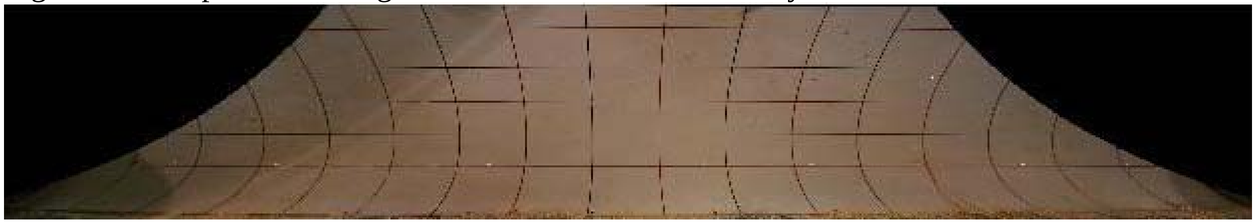


Figure 6. Non-preferred Image of Small Basin Grid with Polynomial 2nd Order Transformation.



To do the polynomial transformation type:
>>Tformpoly=cp2tform([X Y],[Xtran Ytran],’polynomial’); <enter>
By default, Matlab chooses a 3rd order polynomial when no number is assigned.
To do a second order polynomial transformation:
>>Tformpoly2=cp2tform([X Y],[Xtran Ytran],’polynomial’,2); <enter>
To do a fourth order polynomial transformation:
>>Tformpoly4=cp2tform([X Y],[Xtran Ytran],’polynomial’,4); <enter>
(See more in Matlab help under Types of Supported Transformations; see attached.)

Load the image of your test grid into Matlab to test the transformation function you just created. The goal is to make your image look like the transformed grid created in Excel earlier.

To load an image:
>>im=imread (‘”image file name here”.jpg’); <enter>
e.g.: >>im=imread(‘Image_0006.jpg’); <enter>

To view the image:
>>imshow(im) <enter>
Or
>>imview(im) <enter>

To view multiple images in multiple windows:
>>figure,imshow(im) <enter>
>>figure,imshow(“some other image matrix name”) <enter>

Crop the image to be transformed to include only the area of interest. For example, the Small Basin cropped the image around the test grid since there is no node data elsewhere in the image (see Figure 7). Determine which rows you want to keep by looking at the row numbers shown in the imview image viewer window.
>>imcrop=im(1:"#of rows you want selected"**,:,:);** **<enter>**
The left side of the first comma is the series of rows to be selected.
Between the two commas is the series of columns to be selected.
The right side of the second comma refers to the depth of the image.
e.g.: small basin: >>imcrop=im(1:1250,:,:); <enter>
Rows 1 through 1250, all columns, and all depth were selected for the small basin image.
e.g.: big basin: >>imcrop=im(1:1350,:,:); <enter>

Figure 7a. Cropped image of Small Basin test grid that cut out are with no node data.

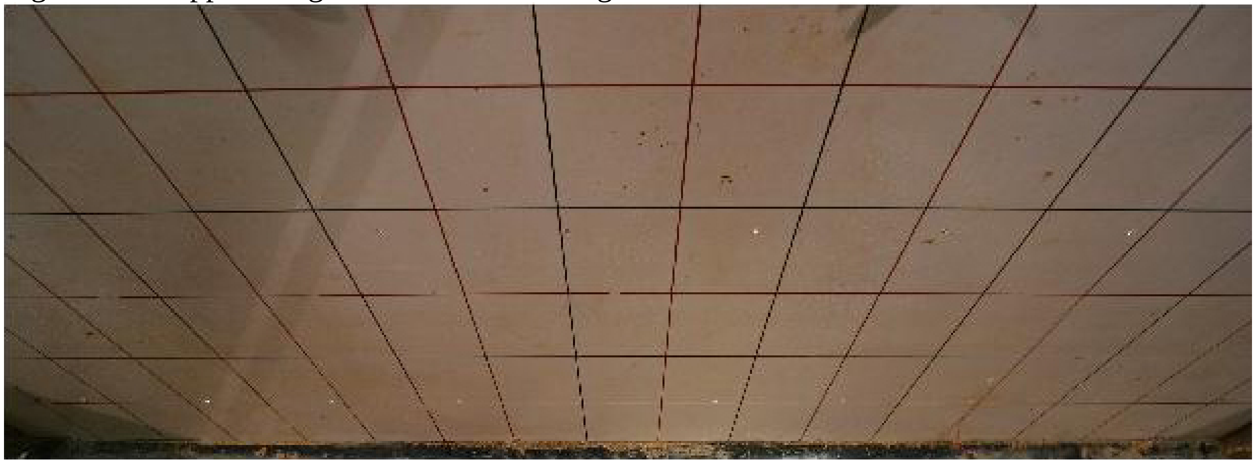
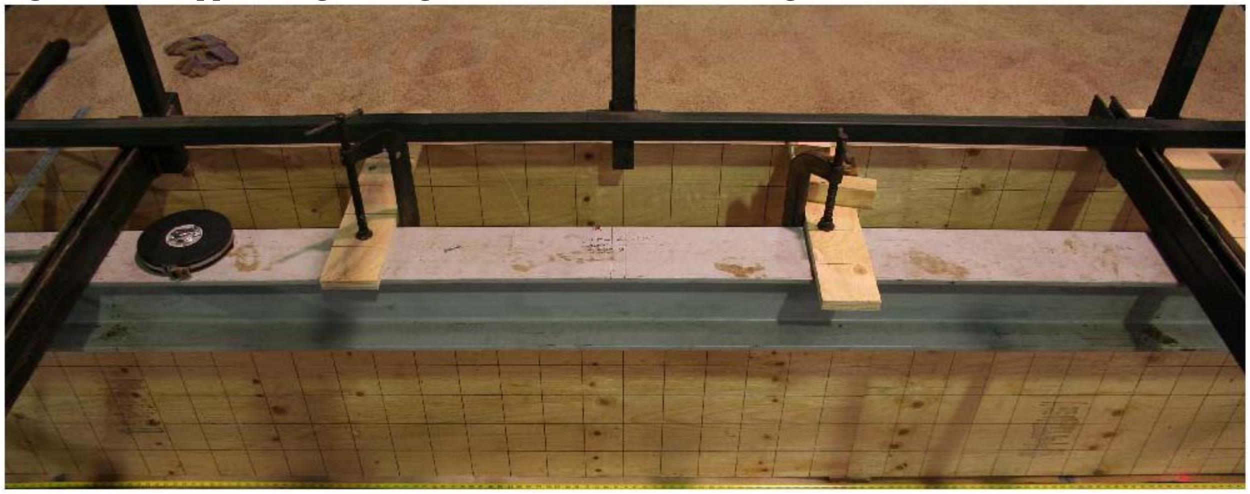


Figure 7b. Cropped image of Big Basin Middle Camera test grid.



Transform the cropped grid image based on the Tform matrix you created:
>>xim=imtransform(imcrop,Tform); **<enter>**
("xim" is an arbitrary name and "Tform" is the arbitrary name of the transformation matrix you created earlier.)

View the image to see the transformed grid image (see Figure 8):

>>imshow(xim);   <enter>

or

>>imview(xim);   <enter>

The transformed image should appear similar to the Excel plot of transformed X and Y nodes done earlier.

Figure 8a. Transformed image of Small Basin test grid image.

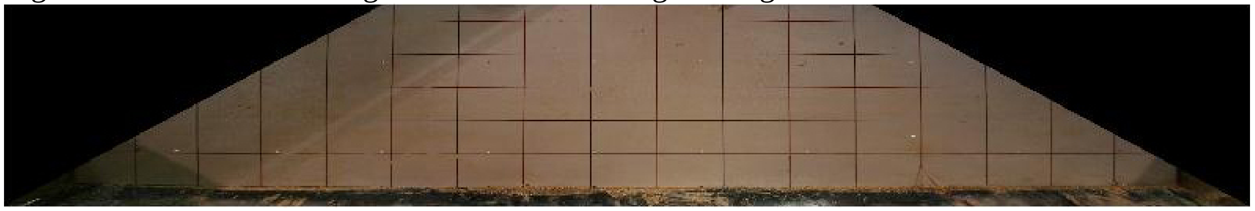
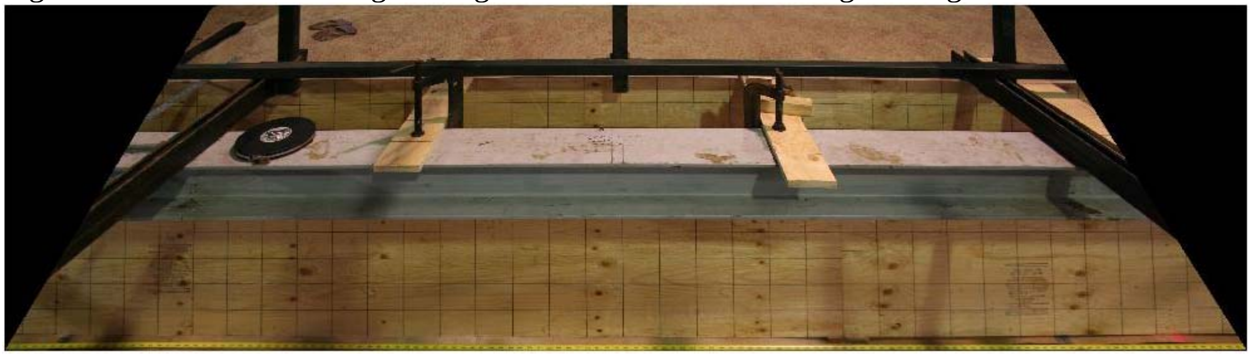


Figure 8b. Transformed image of Big Basin Middle Camera test grid image.



The next steps involve taking an actual image from a survey with the laser line on the bed surface and to process it the same way as done above. The way this image gets processed will determine the methods to process all the images in that specific survey series. Take notes along the way because several steps involve iterative means of finding the best way to process the image.

IMAGE TRANSFORMATION

Load the survey test image into Matlab:

>>imtest=imread('"image file name".jpg');   **<enter>**

>>imtestcrop=imtest(1:1250,:,:);   for small basin   **<enter>**

or **>>imtestcrop=imtest(1:1350,:,:);**   for big basin   **<enter>**

>>ximtest=imtransform(imtestcrop,Tform);   **<enter>**

Figure 9a. Example transformed test image from Small Basin.

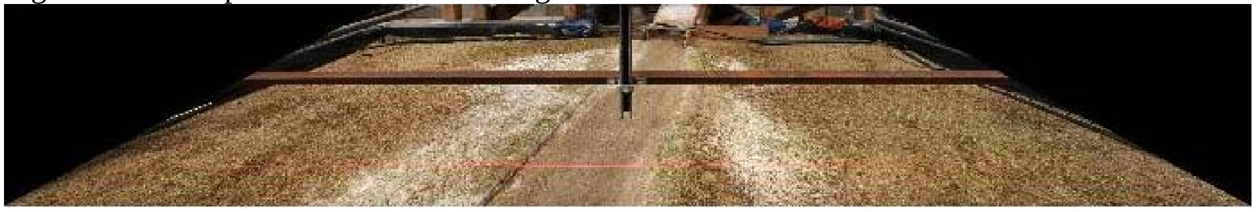
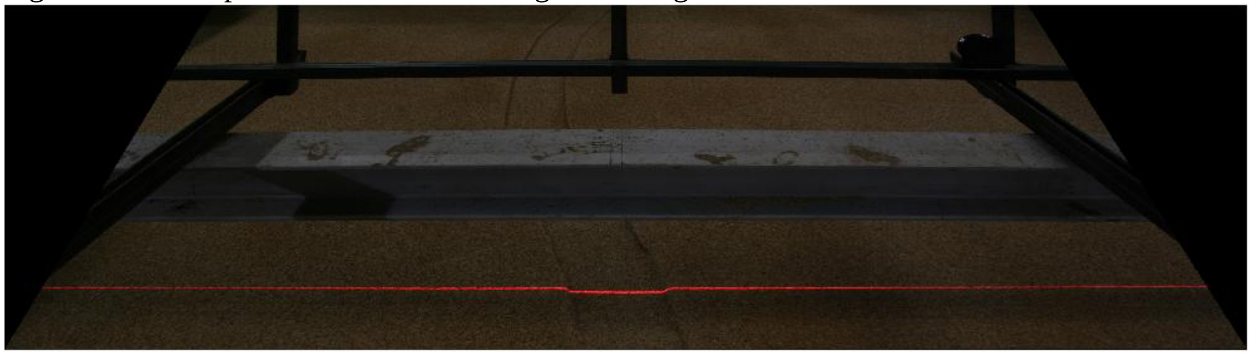


Figure 9b. Example transformed test image from Big Basin Middle Camera.



COLOR FILTERING

Next, the transformed photo is separated into red, green, and blue components and the green is subtracted from the red to isolate the red laser line. First, we need to convert the Uint 8 “ximtest” matrix to a double matrix in order to filter out colors to better detect the laser line:

>>ximtest2w=double(ximtest)./255; <enter>

“double” converts “ximtest” (the transformed grid image) into a double matrix

“.” is required before an operator, such as multiplication or division

“/” is a dividing operator

“255” is the value you are dividing into your matrix

Filter out the colors to enhance the red laser line from the basin. This is an iterative process and may take several attempts before finding the ideal filtering method. Subtracting green from red works best for the Small Basin, Big Basin, and Joel’s experimental surveys (Figure 10).

To subtract green from red:

>>rg=ximtest2w(:,:,1)-ximtest2w(:,:,2); <enter>

The first and second “:” refers to all rows and columns, respectively, and the third “:” refers to the color value. “1”=red, “2”=green, and “3”=blue.

To subtract blue from red if needed:

>>rb=ximtest2w(:,:,1)-ximtest2w(:,:,3); <enter>

Figure 10. Green subtracted from Red color in Small Basin test image.

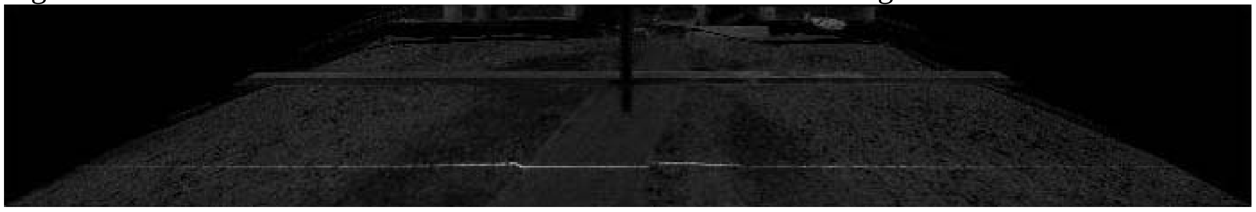
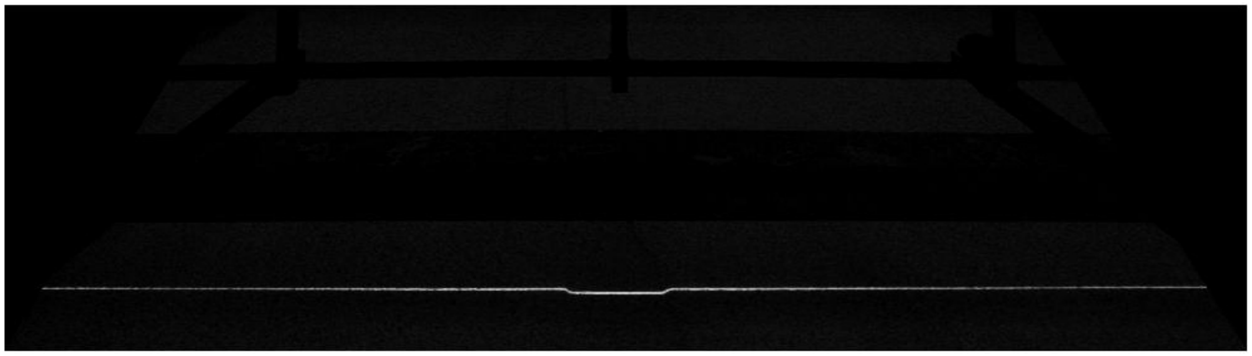


Figure 10b. Green subtracted from Red color in Big Basin Middle Camera test image.



Next, crop the image around the laser line so as to remove surrounding areas that may interfere with the next steps. Again, this is an iterative process, especially if the camera/laser unit on the carriage is not parallel to the basin (e.g.: the Big Basin and Joel's experiments). You need to keep a big enough area around the laser line so that all images taken in a survey will have their laser lines preserved after the cropping. The carriage in the Small Basin is parallel to the floodplain surface so we can crop closely around the laser line.

Since "rg" is a 2D matrix, we only need to define the range of rows and columns to crop. For example in the Small Basin:

```
>>rgcrop=rg(490:520,600:3276);      <enter>
```

We cropped the area between rows 490 and 520 and columns 600 and 3276.

```
>>rgcrop=rg(800:1028,130:3660);    <enter>          for the big basin
```

EDGE DETECTION

Next, Matlab needs to "edge detect" the laser line in the image. This function utilizes a built-in Matlab edge detections algorithm with a user-chosen threshold to find the edges of the laser line. Matlab then generates a matrix of 0s and 1s (edge of laser). The small and big basins, and Joel's experiments use the "Canny" edge detections algorithm.

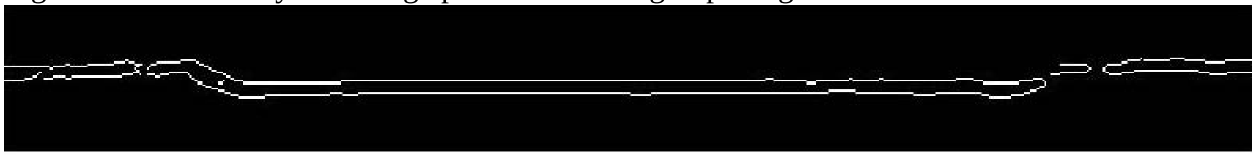
>>line=edge(rgcrop,'canny',"some determined threshold value"**);   <enter>**

The edge command tells Matlab to detect the edges of the pixels that stand out from the image; ie: the laser line.
The 'canny' command is an edge detection filter function that appears to be the best filter available to this function in Matlab. The Canny method finds edges by looking for local maxima of the gradient of I. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be fooled by noise, and more likely to detect true weak edges.
The threshold value to use is up to you based on how the output image appears. The range of thresholds is between 0.1 and 1.0. The lower the number, the more "noise" will be left in the image, but the higher the number may leave too much important data out of the image. The best threshold value will change slightly from survey to survey due to variations in light on the experiment during each survey. You can also tell Matlab to use a range of thresholds, for example:

```
>>line=edge(rgcrop,'canny',[0.3 0.5]);        <enter>
```

The range chosen is between 0.3 and 0.5, but this range can vary based on your needs.

Figure 11. The binary line image produced from rgcrop image with a threshold of 0.4.



ELEVATION MATRIX

Next, the "line" matrix is multiplied with a matrix of predetermined elevations. Each column is then averaged to yield a single elevation for each column. The profile is then saved as a row in a master matrix that contains all the profile for a given bed survey.
First, create the matrix of predetermined elevations. The dimensions of this matrix must be the same as your "line" matrix dimensions. The command below uses an example from the Small Basin, so change the row and column ranges to your specific needs:
>>[X Z]=meshgrid(1:2677,1:31); <enter>

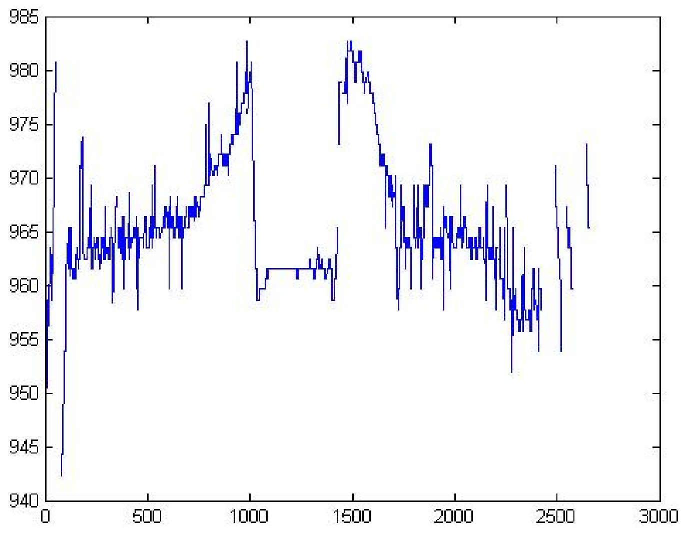
Next, fill "Z" in with the appropriate elevations to get a matrix of elevations. Select an arbitrary max elevation (e.g.: 1000) for the top of your survey photo and then subtract off a pixel increment with each row. Use the pixel height value you determined in Excel for the row increment value; e.g.: the Small Basin vertical pixel height is 1.923077mm.
>>Elev=1000-(Z.*1.923077); <enter> for small basin
>>Elev=1000-(Z.*1.265823); <enter> for big basin

Now multiply the "line" matrix by the "Elev" matrix you just created:
>>Zs=(line.*Elev); <enter>

Each column will have to be averaged using a simple program called "COLMEAN". This program will take a matrix of line edge elevations and determine a single elevation of the line by averaging the value of the edges. Remember that the "line" matrix (Figure 11) had two lines. COLMEAN simply averages these two lines to create just one line per horizontal pixel. It then processes the input matrix "survey" column by column in order to remove 0 from the mean calculation. Make sure the "COLMEAN" program is located in the "Current Directory" window:
>>Zsmean=colmean(survey); <enter>

To view the cross-section you finally created:
>>figure,plot(surveyel) <enter>

Figure 12. Example plotted cross-section of a single survey image from the Small Basin. Note the "noise" and broken line due to vegetation interference and steep banks. The vertical axis, Z, is scaled to millimeters while the horizontal axis, X, is arbitrarily scaled by Matlab.



BATCH PROCESS ENTIRE SURVEY

Add Slope to Small Basin Surveys:

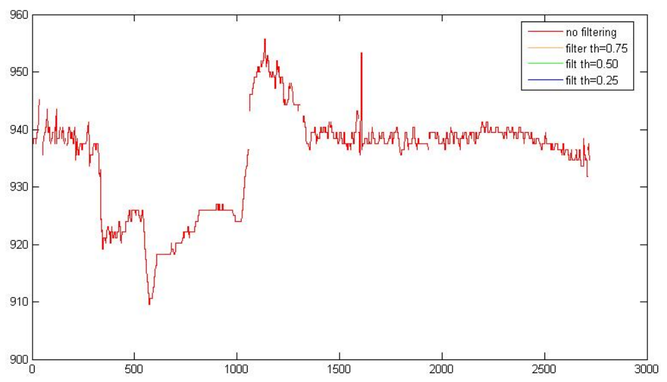
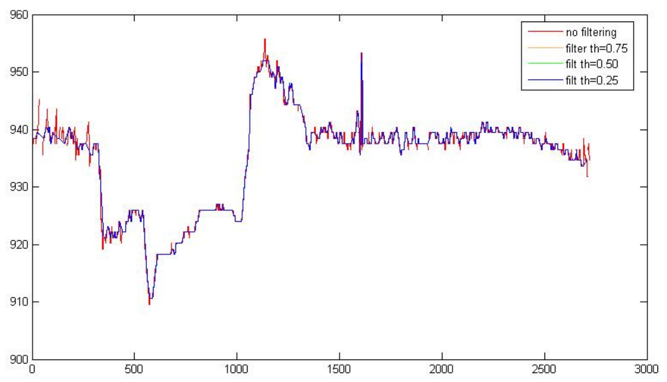
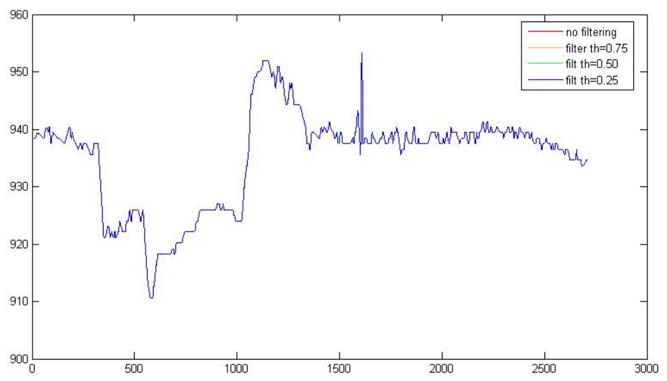
>>slope=0.5*(0:106);	"0.5" is for 0.5mm vertical rise drop over 50mm interval
>>slope=slope';	swaps rows and columns (1x107)$\rightarrow$(107x1)
>>for i=1:2721, bed(:,i)=slope;,end	

Test this by looking at portions of the bed matrix:

>>bed(:,2005)	all rows, column 2005: should be 0-53 with 0.5 increments
>>bed(1,1:20)	for row 1, columns 1-20: should be all zeros
>>bed(2,1:20)	for row 2, columns 1-20: should be all 0.5
>>surv=survey111705step3-bed;	subtract bed surface from survey surface
>>surf(surv)	view figure of survey surface with slope incorporated
>>save bed bed	save bed surface matrix to be used again for other surveys

Noise Cleaners / Survey Line or Gap Filtering:

Using 'slopefilt_final.m':
Copy file into survey's folder so it can appear in the current directory window.
Load survey matrix file into MATLAB so it appears in the workspace window.
>>[filtered]=slopefilt_final(input,thresh);
"filtered"= name of output matrix file
"input"=name of survey matrix file you want to filter
"thresh"=the threshold value for the amount of filtering; choose between 0.01-1.0
example:
>>[survey011006step5slopefilt25]=slopefilt_final(survey011006step5,0.25);
To view the filtered survey surface:
>>figure,surf(survey011006step5slopefilt25);

	No Slope Filtering. Can see noisy peaks on floodplain and gaps along steep banks
	Overlap of cross-sections: red line has no filtering and blue line has filtering with a threshold of 0.25.
	Slope filtered cross-section with a threshold of 0.25. The number of peaks is less and no gaps exist.

Big Basin Middle Camera Initial Test Process (1/1/06):

```
load('BB_Mid_XY.txt');
load('BB_Mid_transXY.txt');
X=BB_Mid_XY(1:239,1);
Y=BB_Mid_XY(1:239,2);
Xtran=BB_Mid_transXY(1:239,1);
Ytran=BB_Mid_transXY(1:239,2);
TformBBMid=cp2tform([X Y],[Xtran Ytran],'projective');
im=imread('Image_0417.JPG');
imshow(im)
imview(im)
imcrop=im(1:1320,:,:);
imview(imcrop)
imcrop=im(1:1350,:,:);
imview(imcrop)
%-- 1/1/06  7:16 PM --%
imview(imcrop)
imshow(imcrop)
xim=imtransform(im,TformBBMid);
imview(xim)
%-- 1/1/06  7:34 PM --%
load('C:\Documents and Settings\Glen Leverich\My Documents\Geology Academics\Stillwater Sciences\Stillwater Pictures\Surveys\Big Basin Surveys\092705 new basin calibrate\Middle Camera\MidCamCalibrate.mat')
imview(xim)
imshow(xim)
```

xim=imtransform(imcrop,TformBBMid);

```
imshow(xim)
imtest=imread('IMG_0500.JPG');
imtestcrop=imtest(1:1350,:,:);
ximtest=imtransform(imtestcrop,TformBBMid);
imshow(ximtest)
im2=imread('IMG_0471.jpg');
im3=imread('IMG_0627.jpg');
imview(im2)
imview(im3)
%-- 1/1/06  9:43 PM --%
load('C:\Documents and Settings\Glen Leverich\My Documents\Geology Academics\Stillwater Sciences\Stillwater Pictures\Surveys\Big Basin Surveys\102105 New Basin Initial Survey\102105 survey Middle\Test.mat')
imview(im3)
ximtest2w=double(ximtest)./255;
rgtest=ximtest2w(:,:,1)-ximtest2w(:,:,2);
imshow(rgtest)
imview(rgtest)
rgcrop=rgtest(1100:1350,120:3700);
imview(rgtest)
```

```
im2=imread('IMG_0471.jpg');
im3=imread('IMG_0627.jpg');
imcrop2=im2(1:1320,:,:);
imcrop3=im3(1:1320,:,:);
xim2=imtransform(im2,TformBBMid);
xim3=imtransform(im3,TformBBMid);
xim2w2=double(xim2)./255;
%-- 1/1/06 10:07 PM --%
xim2w2=double(xim2)./255;
imshow(xim2)
im2=imread('IMG_0471.jpg');
im3=imread('IMG_0627.jpg');
imcrop3=im3(1:1320,:,:);
imcrop2=im2(1:1320,:,:);
xim2=imtransform(imcrop2,TformBBMid);
xim3=imtransform(imcrop3,TformBBMid);
xim2w2=double(xim2)./255;
xim2w3=double(xim3)./255;
rg2=xim2w2(:,:,1)-xim2w2(:,:,2);
rg3=xim2w3(:,:,1)-xim2w3(:,:,2);
imview(rg2)
imview(rg3)
rgcrop=rgtest(800:1028,130:3660);
imview(rgcrop)
line=edge(rgcrop,'canny',0.2);
imview(line)
[X Z]=meshgrid(1:3531,1:229);
ElevBBMid=1000-(Z.*1.265823);
ZsBBMid=(line.*Elev);
ZsBBMid=(line.*ElevBBMid);
ZsmeanBBMid=colmean(ZsBBMid);
figure,plot(ZsmeanBBMid)
```