## Optical Odometry Algorithm Evaluation

Experiments involving position shift extraction from small sample sets of simulated and real road sequential images using a phase correlation method

Kenneth Laws

10/10/2017

## Algorithm Evaluation

Design or select algorithm to reduce raw data to desired data products.  
Test and evaluate different methods,

best data analysis algorithm/strategy for prototype experiments selected

10/13/2017 – researching algorithms appropriate to image matching

One topic of relevance to this project is image registration. In image registration, a source image is transformed to match a target image. Techniques include intensity-based and feature-based. Transformation modes should be limited to rotation and translation. Frequency domain methods can work for simple translation and rotation, applying a phase correlation to a pair of images. Frequency domain methods operate on the FFT of the images and result in large performance advantages.

Spatial Algorithms to investigate include:

* Fisher Vector
* VLAD
* SIFT
* MSER
* k-means
* hierarchical k-means
* agglomerative information bottleneck
* SLIC superpixels
* quick shift superpixels
* large scale SVM training
* others

Frequency domain algorithms to investigate:

* Phase correlation in frequency domain

10/16/2017

*Phase Correlation in Frequency Domain*

This technique has risen to the top as a candidate algorithm because of its relative simplicity and the need for a very fast algorithm. This method leverages the power and speed of the fast Fourier transform to solve the registration problem.

The way the method is applied to this problem is as follows. We are given two images, sequential in time, that may be shifted by some unknown amount in the x (y) dimension, defined perpendicular (parallel) to the motion of the vehicle when not turning. The images may also be rotated, but for now we assume either no, or insignificant rotation. We define a rectangular region on the first image and refer to this region as the template. We locate the template near the bottom center of the image, defined with the vehicle forward direction of motion down. This template image section is denoted *gb*. The entire region of the second image (or some sub-region, if information can be used to narrow the search such as approximate vehicle speed) is denoted *ga*. The objective is to locate image section *gb* on image *ga*. To do this we use the Phase Correlation method (with normalized cross-power spectrum) suggested by Kuglin and Hines.

Kuglin, C. D. and Hines, D. C., 1975. The phase correlation image alignment method. *Proceeding of IEEE International Conference on Cybernetics and Society*, pp. 163-165, New York, NY, USA.

We compute the 2D Fourier transform of both images,

,

and compute the cross-power spectrum

where is the Hadamard product (element-wise product) and the absolute value, ||, is taken element-wise as well. That is, the product of the Fourier transforms of x and y is equal to the Fourier transform of the cross-correlation function of x and y (convolution theorem). The Fourier transform of the cross-correlation function is element-wise normalized to have unity magnitude so that the result after taking the inverse Fourier transform is the normalized cross-correlation.

We then take the inverse Fourier transform of the normalized cross-power spectrum

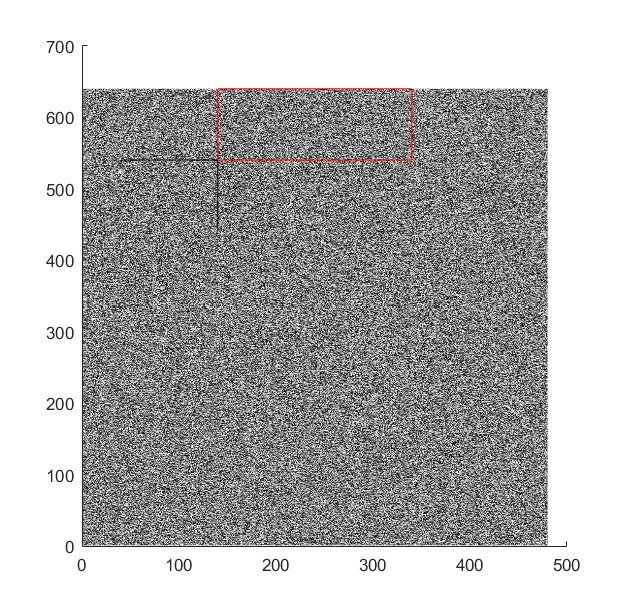
,

and obtain the normalized cross-correlation, *r*. The peak value of *r* gives the location of the lower left corner of the region of image *ga* (target region) that matches the template region.

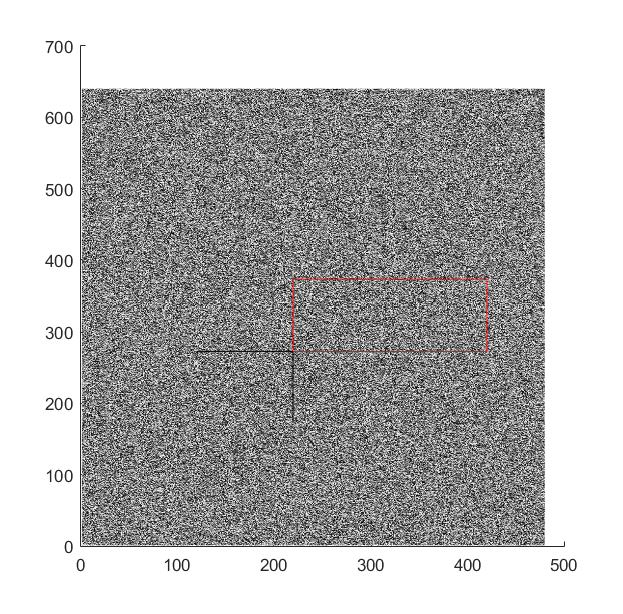
Have written a matlab script to test image matching using phase correlation in frequency domain. (feasibility3.m)

Works very well with simulated data. In the simulation, 2 random images are generated, the second one has some overlap with the first and is shifted. To use nomenclature used for phase correlation image registration, the first image contains the template that is attempted to locate on the second image (background). The template is defined as the area defined by the red square in the template image and then the template matchup is found on the background image.

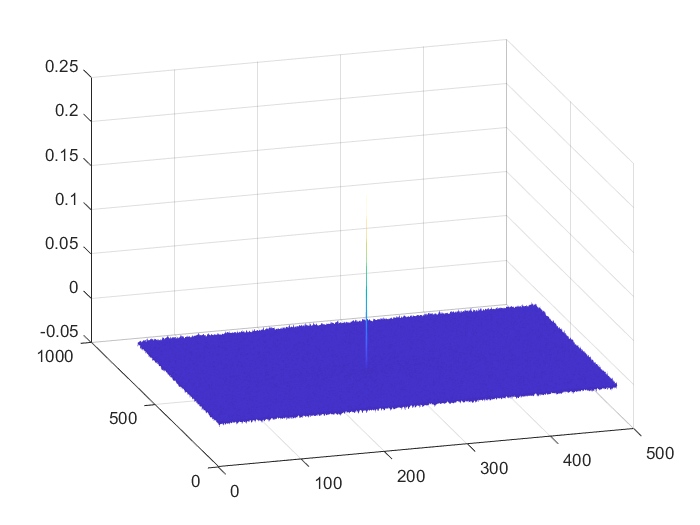
In the simulation, a registration mark is placed on the template image and on the background image, at the location of the template area, lower left corner.



Simulated template image with registration mark (intersection of black lines) and the template region (red rectangle).



Simulated background image with retrieved location of the template region (red rectangle) and expected location of lower left corner indicated by registration marks (black lines).



Plot of phase correlation showing very sharp peak at the template location.

Results of above simulation analysis:

subframe size: 200 x 100

template lower left corner position in template image: (140, 540)

simulated vehicle speed: (vy: 50 mph, vx: -15 mph)

distance travelled between frames (dy: 0.1315 m, dx: -0.0394 m)

distance travelled between frames (dy: 267 pix, dx: -80 pix)

position of template in background image: (220,273 )

retrieved template position (220,273 )

analysis took 5.48E-02 sec

## Analysis of phase correlation algorithm performance using actual road images

## Driveway 1

Have collected a small data set of sample data using FLIR camera mounted to a rolling cart.

Lens: '3.5mm 1:1.8 ½.5”IR MP-5

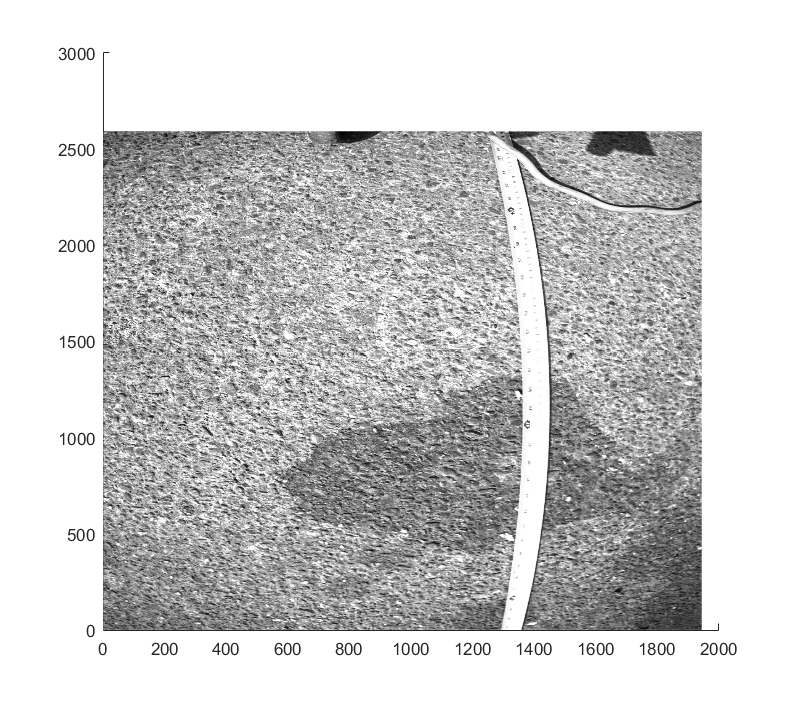
Camera: BFLY-PGE-50A2C-CS

Images are of roadway out front of warehouse and include a tape measure in the frame for manual registration. The first set of images are blurry due to poor focus and show a significant amount of distortion due to lens angle. Using the same subframe size as was used for simulation experiments, the algorithm fails on the real data set.

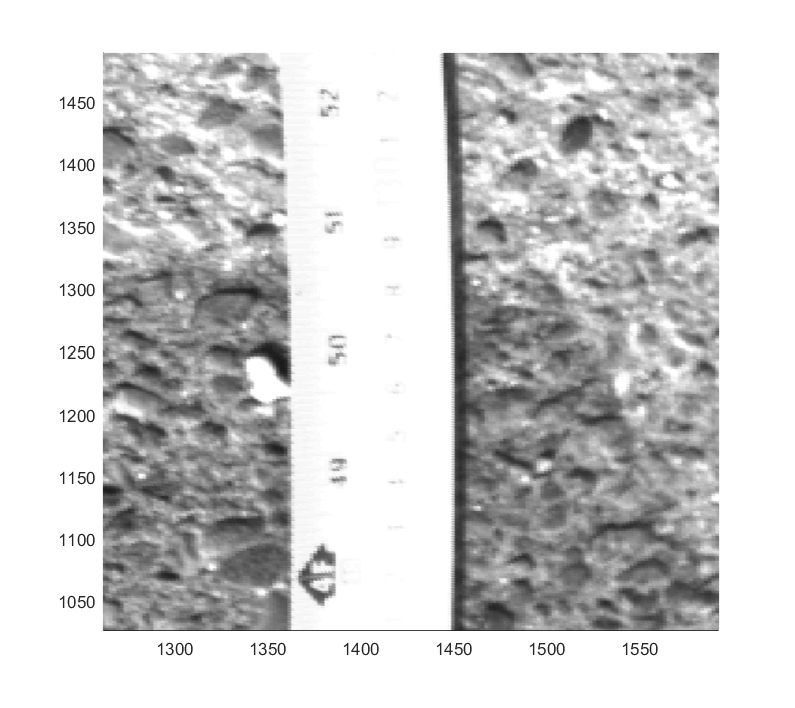
Initial size of the subframe was 200 x 100. I am finding some improvement in results with larger subframe size (800 x 400). Further increasing to (900 x 400) no further improvement.

Results are very poor with uncertainty in phase correlation looking very high. Trying normalizing the images before processing.

Discovered that the image data is being stored by matlab as uint8. This might affect operations to normalize and possibly the fft operations. Try converting to float. – No improvement



Sample road image showing distortion due to lens



Same image, section showing focus and resolution.

Analysis: A subframe is defined at a given position, red square in image 1. Using phase correlation method, the best solution for locating the lower left corner of the subframe is found.

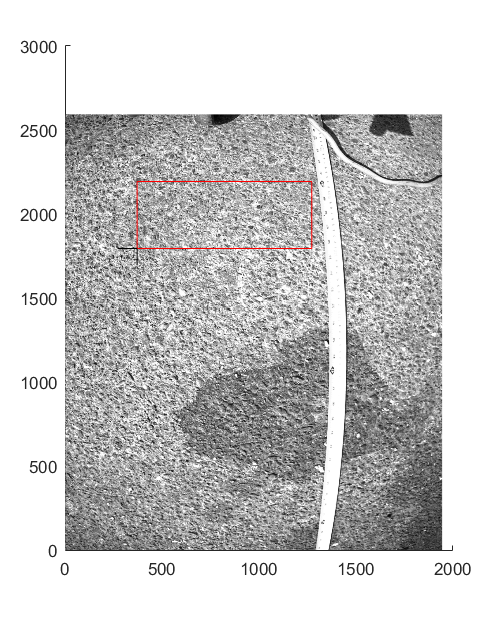


Image 1, template image

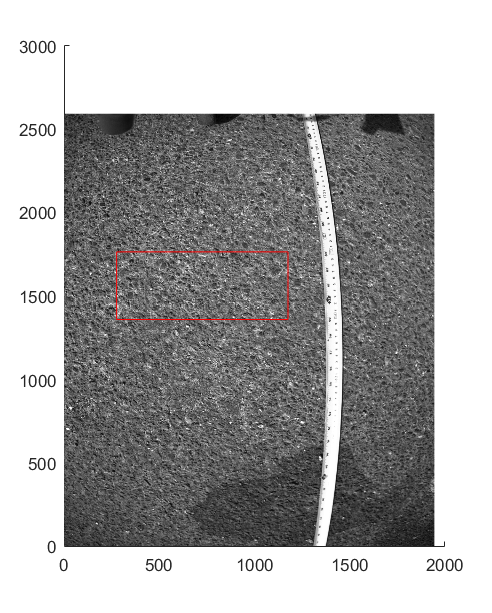
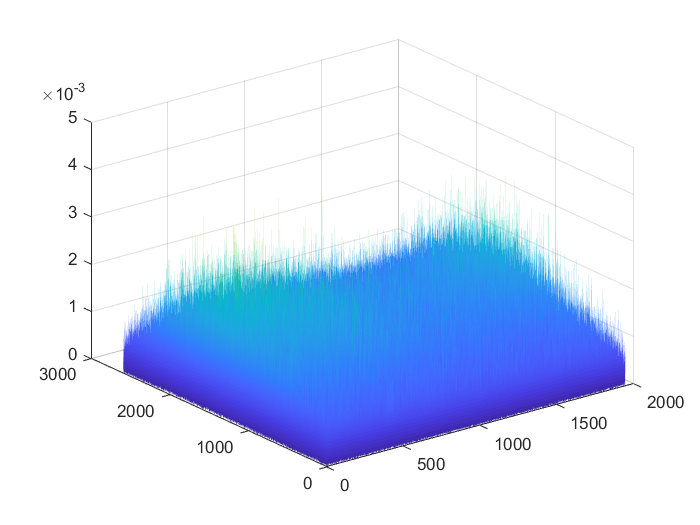


Image 2, background image

The template is taken as the area defined by the red square in the template image and then the template matchup is found on the background image. The location of the template in this example is found at a shift of 430 pix, down, 96 pix left.



Phase correlation as a function of position of the template on the background. Surface does show weak maxima mostly dominated by noise.

The shift of the image can be estimated using the tape measure included in the frame, top of the frame is about 67” in image 1 and about 74.5” in frame 2. The error in vertical position of the subframe (red box) is about 60” – 63” = 3”, or about 50% of the measurement.

10/17/2017

### Driveway 2

Switched cameras and lenses, going to a narrower angle lens.

Camera: BLFY-PGE-20E4C-CS

Lens: 8MM 1/1.8 ir mp

Data runs:

Cal – calibration, lens height: 37 cm, focus camera, measure field of view

Run1 – smooth concrete – shade

Run2 – rough chip seal – shade

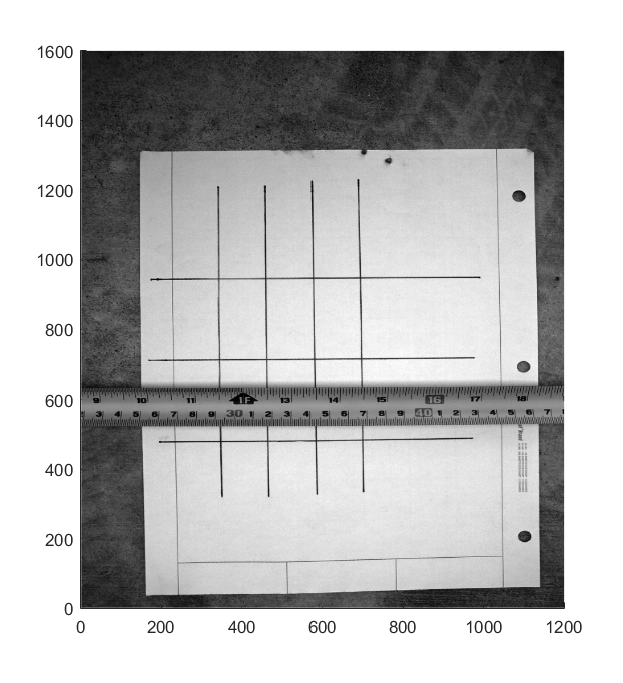
Run3 – rough chip seal – sun

Cal2 – change height to 60 cm, focus camera, measure field of view

Run4 – rough chip seal, sun

Run5 – smooth concrete, sun

Estimating image resolution using roadTest2\_cal.m



Sample image showing focus and resolution

Image size is 1600 x 1200

Physical size estimated by measurement is 51 - 16.5 = 34.5 cm, 47.6 – 22 = 25.6 cm.

Camera height was 37 cm.

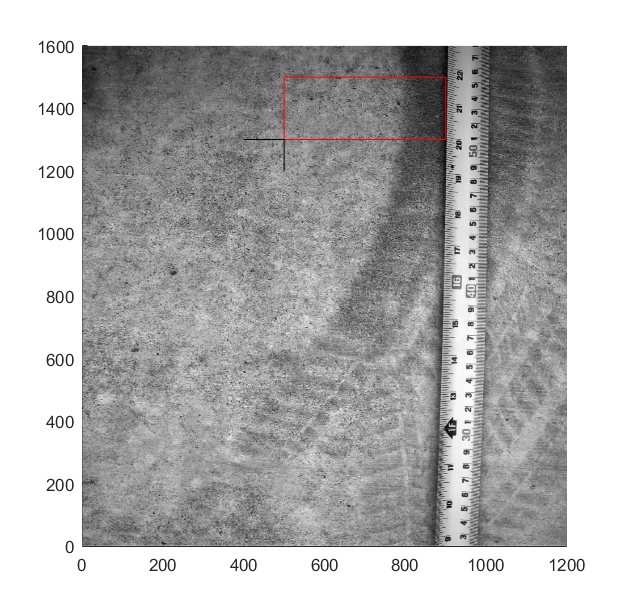
Resolution in pixels per meter:

1600 / .345 = 4638 pix/m x 1200 / .256 = 4688 pix/m

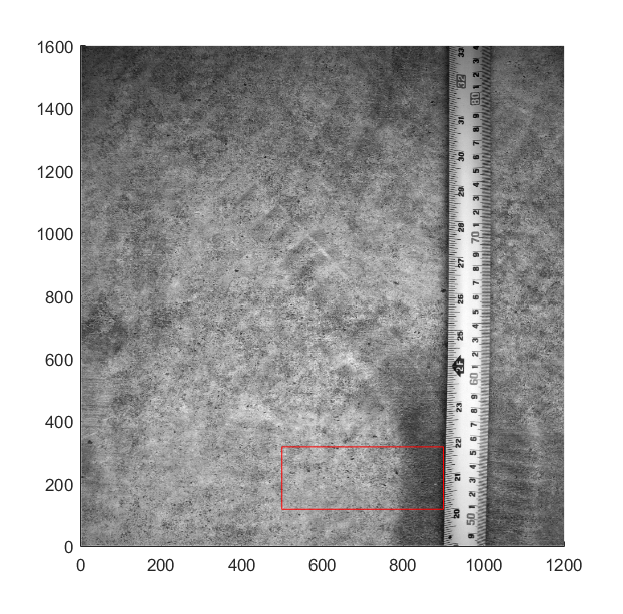
4638 pix/m x 4688 pix/m

Results with the second camera and lens setup are shown below.

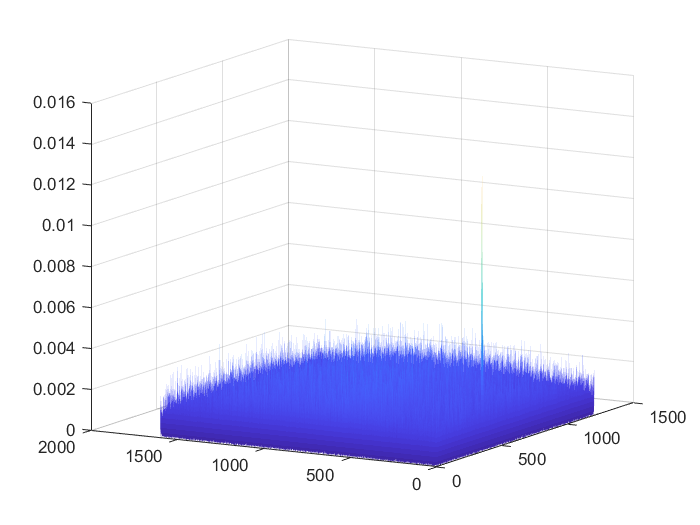
### Run 1 results



Template image with location of template area set so that position of the template can be accurately estimated from the tape measure in the image. The lower right corner of the template area is at about 19.9 cm. The right edge of the template area is within 1 mm of the tape.



Background image with retrieved location of the template area. Retrieved template lower corner is at about 20.05 cm and again is within about 0.5 mm of the tape. Vertical accuracy of the position retrieval is estimated at within 2 mm vertically and within 1 mm horizontally.

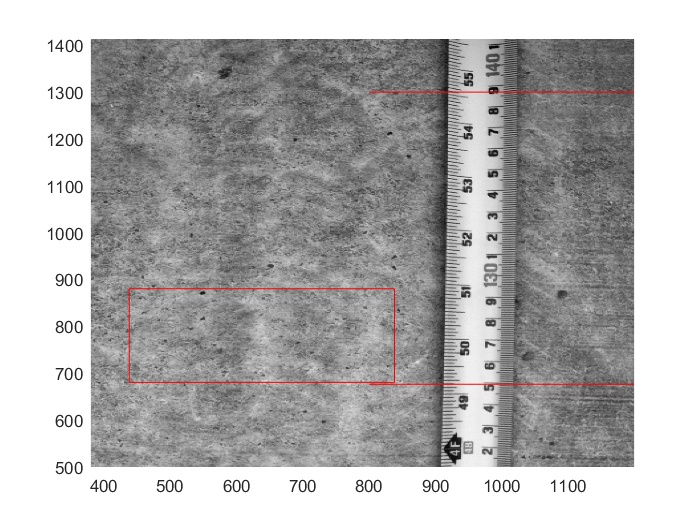


Phase correlation in frequency domain from above images. Unlike the driveway 1 experiments, this correlation plot shows a sharp peak well separated from the noise.

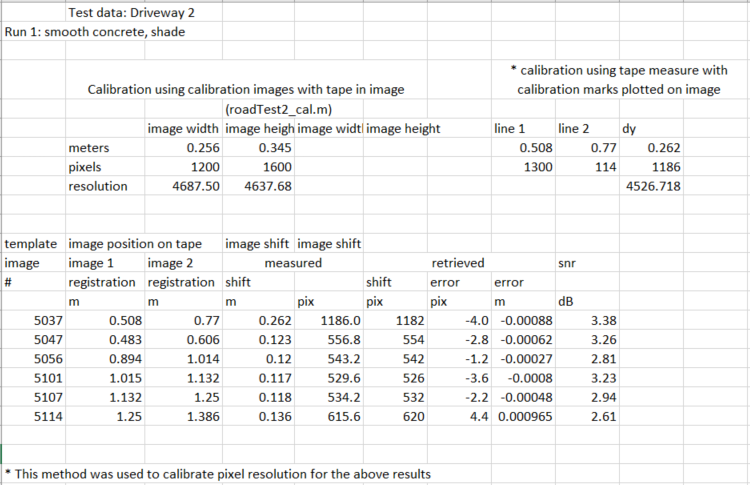
### Delta y retrieval

The y axis is defined as the axis with larger number of pixels and is approximately parallel to the direction that the cart was moved during each shift, roughly parallel to the measuring tape laid along the ground. Experiments show a high degree of accuracy retrieving change in position, but this accuracy relies on an accurate estimate of the conversion factor (image resolution in pixels per inch) used to convert shift in pixels to shift in meters. It is likely that an additional sensor will be needed to correct for changes in the distance from camera to road that affect this conversion factor.

Note also that a quality factor could be extracted from the statistical distribution of the correlation values.

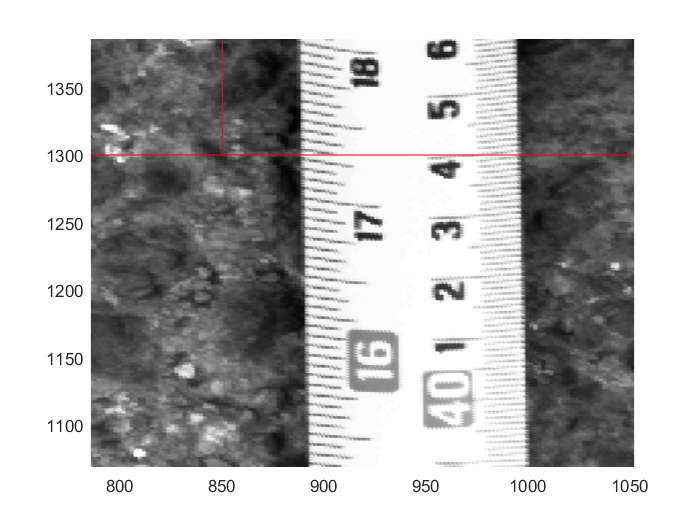


Sample image showing registration lines used to measure location of the retrieved template, lower box and red line, and the shift of the image from the previous image, upper red line position is measured in both previous and current image.



### Run 2 Results

Same setup as run 1 but with rough chip seal road surface on street in front of warehouse, shade



Focus looks less sharp than in run 1 images, still sharp enough for mm resolution on tape measure.

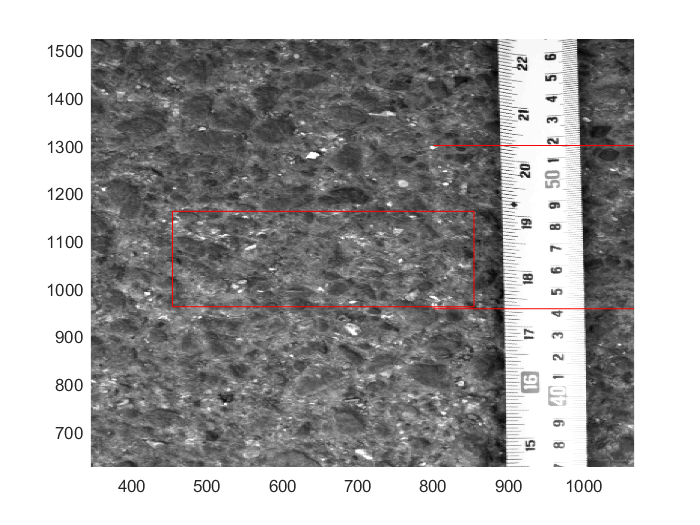
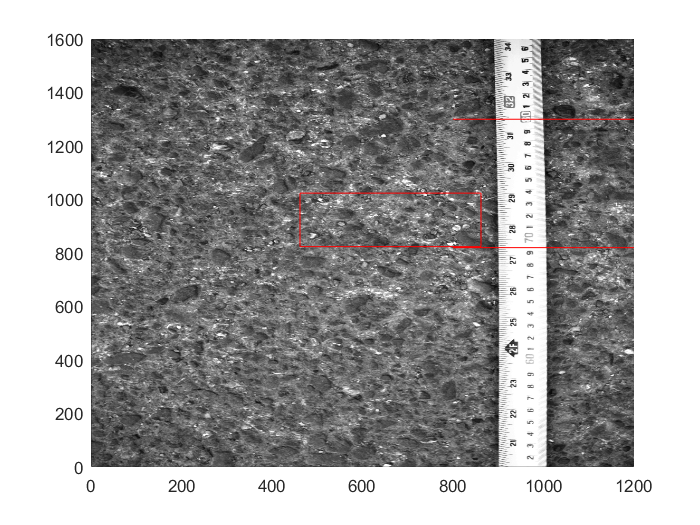


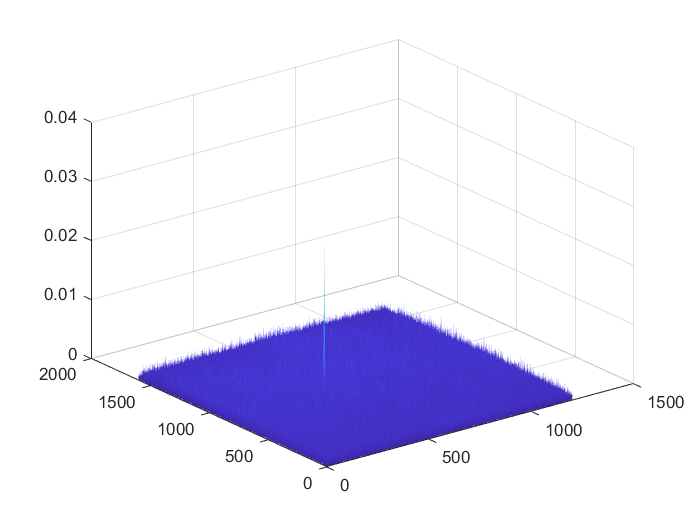
Image 095343 used for calibration by measuring distance between red lines on tape both in pixels and meters

Previous calibration value used in run1 = 4526.72 pixels/meter

Calibration using this image = 4506.67

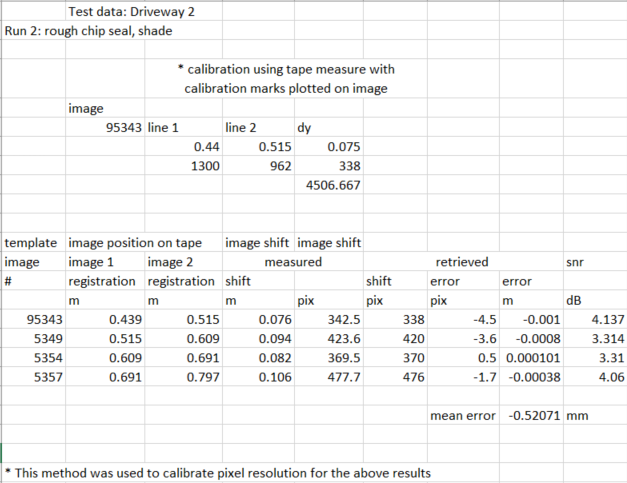


Sample image 095357 showing location of template on background with registration lines (red). Measured image shift is based on location of upper red line in this and the previous image. Upper red line shows position of the template on the previous image.



Correlation plot showing sharp spike at template location. Snr for this case was 4.06 dB (better than run 1 results with smooth concrete.

Run 2 Results



### Run 3 Results

Same setup as run 2, rough chip seal road surface on street in front of warehouse, but in sunny conditions. A Quick look at one image pair showed good results similar to run2 and 3.68 dB. This analysis left out in interest of time.

### Run 4 Results

Same setup as run 3, but with camera height raised to 60 cm , rough chip seal road surface on street in front of warehouse, sunny conditions. Length of the image: 185.8 - 128.5 cm = 57.3 cm

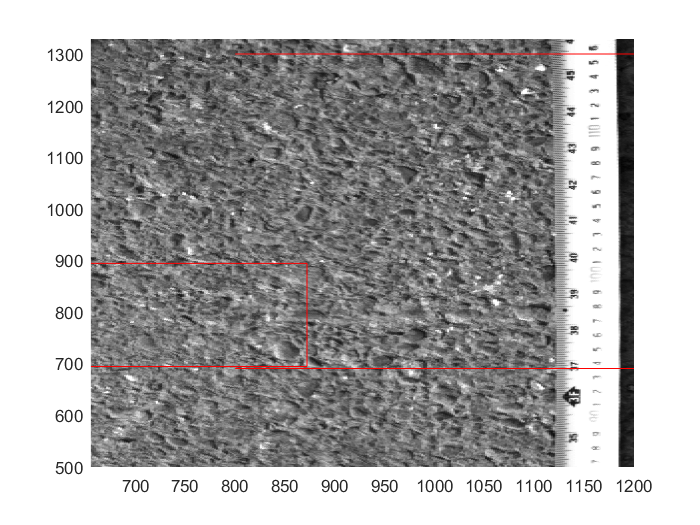


Image 100436, used for calibration: glare of sun on tape measure reduces accuracy of calibration and distance measurement in general. Estimate of calibration from this image = 2754.5 pix/m

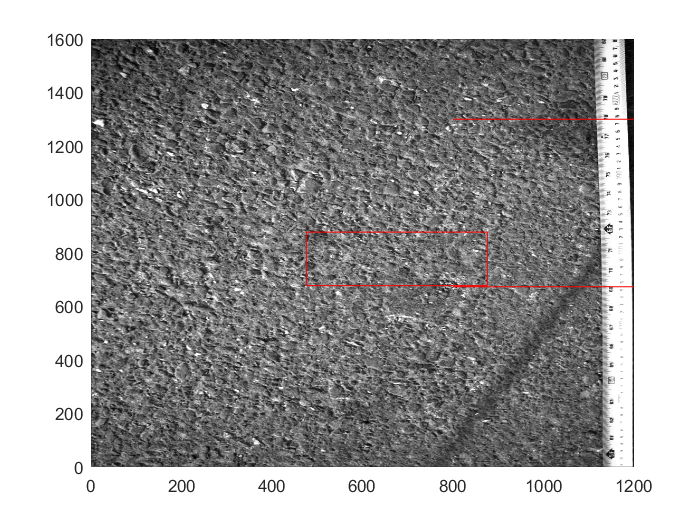
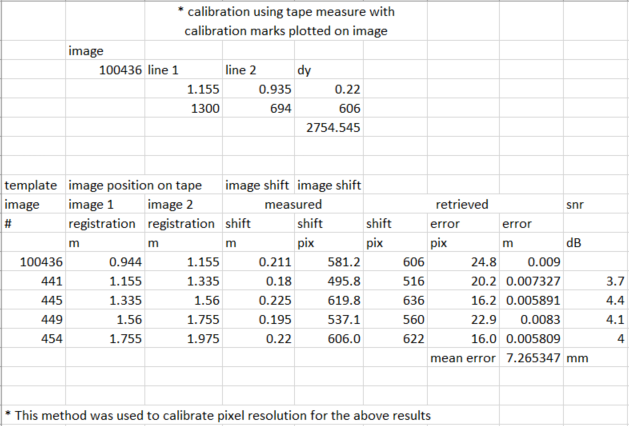


Image 100454, example showing retrieved location of template (red box) and registration lines for distance measurement.

Run 4 results



Results for run 4 (60 cm camera height) are more than a factor of 2 larger than the 37 cm height results, however, error is consistently about positive (5 to 8 mm) probably indicating calibration error rather than random error.

## Examination of SIFT Algorithm

Implementations of SIFT are available for Matlab, but all the ones I found require the signal processing toolbox. It might be worth getting this toolbox for this or later projects.

\*\*\*\*This program finds the key points and their descriptors of two images and matches them

\*\*\*\*\*\*\*\*Output of this program will give you the matching percentage of key points

\*\*\*\*\*Submitted to Texas Instruments on October 1st 2013

\*\*\*\*\*\*\*By:Ch.Naveen\*\*\*\*\*\*\*\*

Requires image processing toolbox

Matlab implementation.

%this code is the Matlab implimentation of David G. Lowe,

%"Distinctive image features from scale-invariant keypoints,"

%International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

A class of algorithms, which may be used to determine similarity in a far more efficient manner than methods currently in use, is introduced in this paper.   
*A Class of Algorithms for Fast Digital Image Registration*. Available from: <https://www.researchgate.net/publication/224483144_A_Class_of_Algorithms_for_Fast_Digital_Image_Registration> [accessed Oct 20 2017].

Have been able to get a C++ version of sift compiled and running on linux box. Have generated key files for several of the images in the driveway test. These files are very large (12 MB) and take a long time to generate. They would then require processing to retrieve position change. It is very likely that other parameter settings would result in faster generation of smaller files but it still seems likely that the sift approach, or any feature parameterization will require more processing power than phase correlation. Also, given that the images are of roadway, I expect the features to be very small in scale and large in number. This might lend itself more toward correlation than feature identification.

Advantages to SIFT algorithm include invariance to scale, rotation, illumination and noise. Cost of the algorithm is in compute time. Only a small number of key point are needed for a matchup, but it seems that a full set of key points must be generated for the background image. Also, for images like ours, many key points on the template will not find a valid matchup on the background. Therefore, a large number of key points will still need to be generated for the template as well as the background and many will need to be tested to find the small number needed for a match.

## Summary

Position change measurement using sequential image with phase correlation in frequency domain was examined. A small set of images were taken using a camera mounted on a cart. Results were robust over the small sample size witih about 0.5 mm mean error obtained with a 37 cm camera height and 7 mm error obtained at 60 cm height. Part of the cause for larger error with the higher camera was difficulty reading the position from the tape in the images due to glare. Accuracy of a system will depend on accurate knowledge of camera height and inclusion of an additional sensor to measure height in real time + method to calibrate resolution in pix/m based on camera will be an asset.

Examination of SIFT algorithm indicates that it will be difficult and time consuming to find out if optimization of the algorithm will achieve processing load we can deal with. Initial investigations indicate that while it is likely more robust and versatile than phase correlation, it is also much more compute intensive. Phase correlation looks like the best direction for this set of proof-of-concept experiments.

The desired processing speed of the algorithm is as fast as possible, up to the limit of the camera or other hardware. The faster the algorithm, the closer the camera can be to the ground. Practical limitations on camera frame rate though cap the speed requirement at about 160 frames per second. There are faster cameras but there are also limits on speed of other components and on minimum resolution. For a frame rate of about 150 frames per second, we need to process a frame in about 0.007 seconds. Current Matlab code can process an image pair in 0.2 seconds (for the phase correlation in frequency domain algorithm) and abouit 0.1 sec for reading files. The combined time to read two files and process for position shift is about 0.35 seconds and varies a little each time the script is run. In order to process in real time we would need about a factor of 50 improvement in processing time. A quick examination of improvement factors found by people who have used Matlab to develop algorithms and then ported to C/C++ indicated factors of up to 500 are seen, and one source cited a factor of 50 as a rough estimate of average speed improvement.