APPENDIX
Velocity Results

A.1. Observed velocity changes in pixel space

The following results for velocity were appendixed because they required further work to investigate as to why some flow was observed to be negative in nature. The magnitude of displacements at each timestep were subtracted from the initial magnitude. This implied that some positions had change in elevation which corresponded with a 'negative flow' when plotted cumulatively. Because all the data below is only presented as total magnitude of displacement, it does not account for changes in the x or y directions which may be responsible for these observations. In Jones's DIC script, it was found that pixel positions were saved from top to bottom, left to right. This meant that vertical displacements may be in reverse. As such, further corrections are warranted before they can be presented.

A.1.1. Direction of flow

The annually averaged direction of flow at the terminus region was sampled from a GPS station setup on the Tasman Glacier ~ 4 km upstream from the calving face for 2013 to January 2015. The mean direction of flow at this point is used to approximate the flow scale factor (FSF) crucial to undistorting velocity changes on the Tasman Glacier.

For 2013-2015, direction components are given by vX_{mean} = – 3.4e^{-7} m, vY_{mean} = - 8.79e^{-7} m and vZ_{mean} = – 1.349e-7 m.

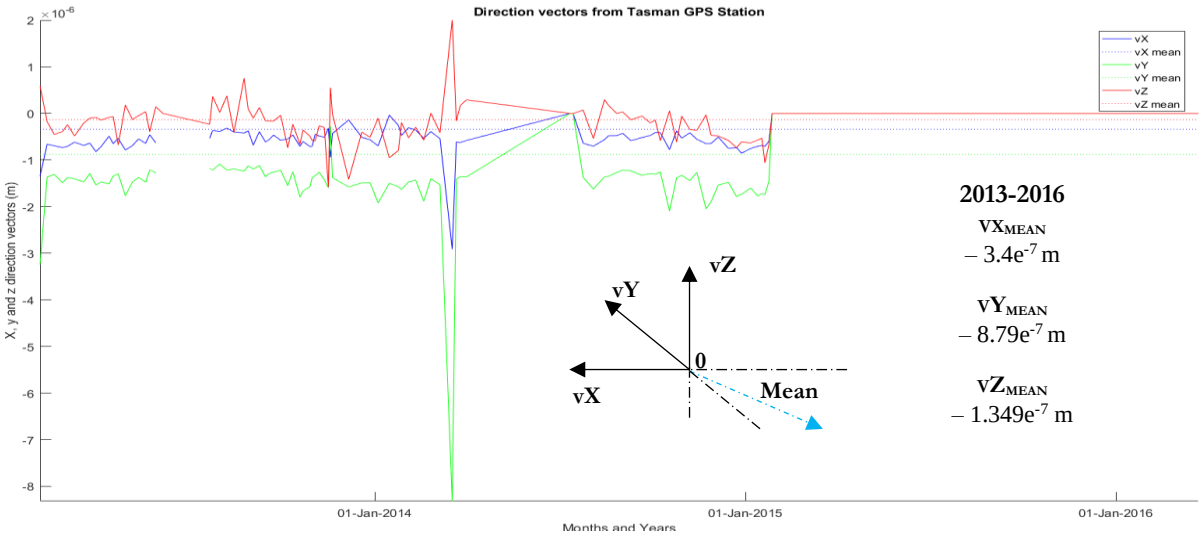


Figure A.1.1. Direction of flow vs time from 2013-2015. Data courtesy of Dr. Huw Horgan.

Using these direction of flow values, the Flow Scale Factor (FSF) and Pixel Scale Factor (or pixel size) was attained at each pixel of every image in our time series. The application of this is seen in the results of *A.1.2*.

A.1.2. Translating correlation outputs to real-world gridded Sample Points

Using the Flow Scale Factors (FSF) for each respective batch, an automated grid created in QGIS was reprojected onto the image plane (see *Figure A.1.2.1)*. This is a crucial step in order to translate the DIC outputs, which had an independently created grid of 40 pixel spacing to real-world 'ground-truthed' sample points.

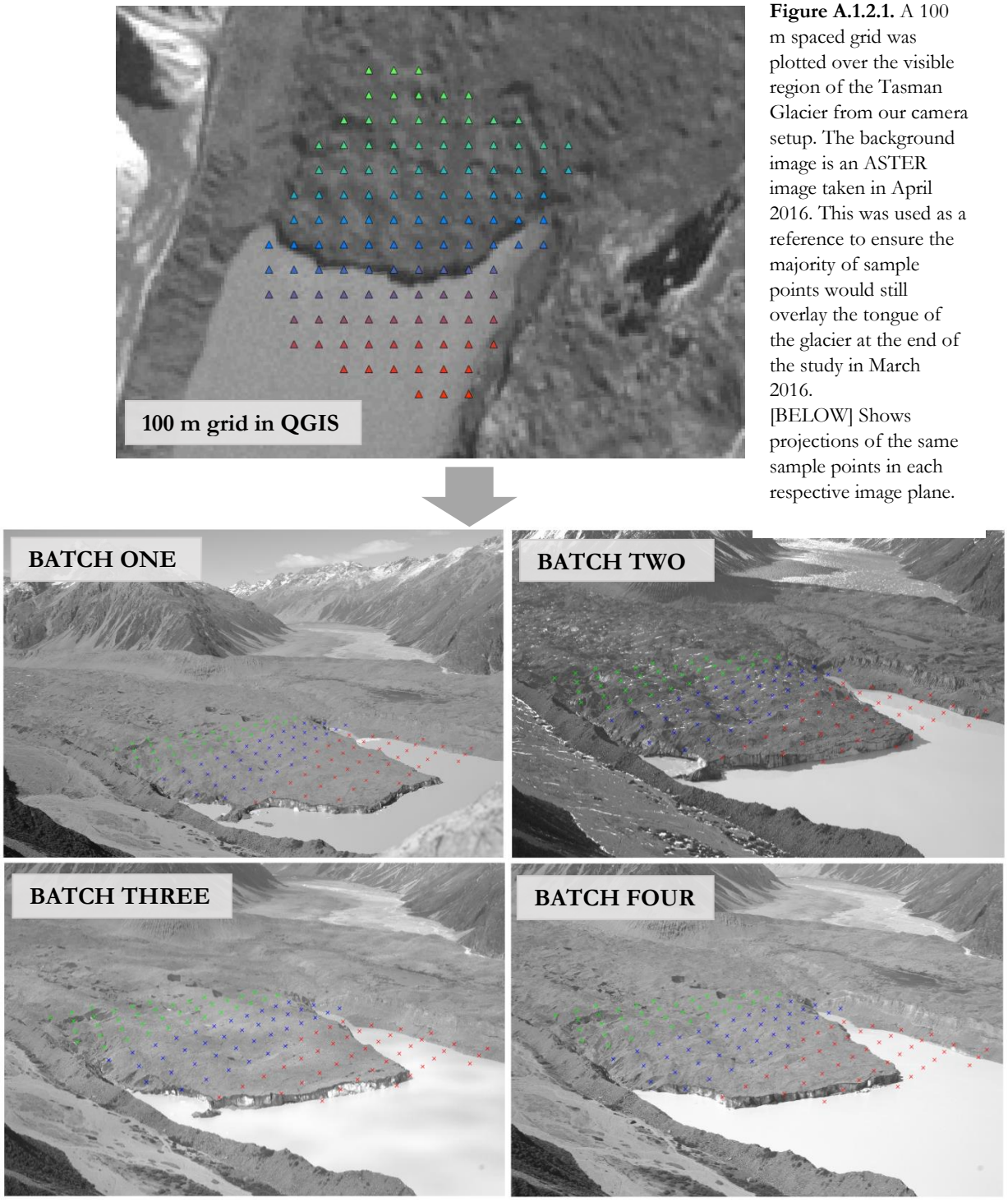
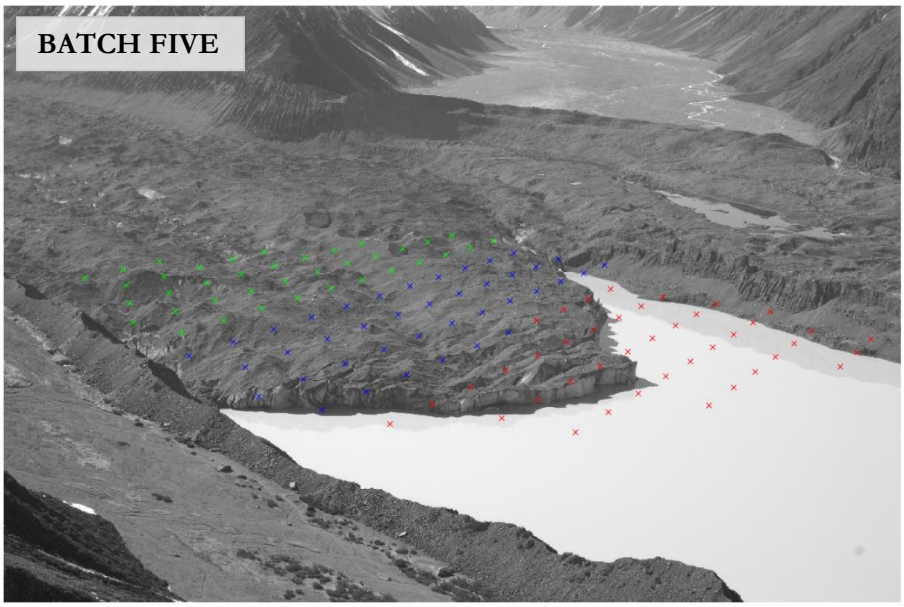


Figure A.1.2.1. A 100 m spaced grid was plotted over the visible region of the Tasman Glacier from our camera setup. The background image is an ASTER image taken in April 2016. This was used as a reference to ensure the majority of sample points would still overlay the tongue of the glacier at the end of the study in March 2016.
[BELOW] Shows projections of the same sample points in each respective image plane.



A.1.3. Final Position and Velocity Results from the Digital Image Correlation (DIC) phase

Position results are plotted cumulatively at each sample point and are presented in five batches in this subchapter. Of the 119 sample points (spaced 100 m apart), three categories (red, blue and green samples) were used to depict relative position changes from the terminating ice cliff.

Figure A.1.3. Breakdown of the 119 samples points used in measurement position and velocity. Approximate distances are measured in QGIS from the MIC at the start of the study to the furthest and closest point in each

Sample Number	**Colour**	**Approximate distance to Main Ice Cliff (MIC) in February 2013**
1 - 39	Green	908 m (nearest) – 1363 m (furthest)
40 – 79	Blue	461 m (nearest) – 808 m (furthest)
80 - 119	Red	0 m (nearest) – 453 m (furthest)

A.1.3.1 Batch One Results

Results from batch one show the most position data (i.e. lines) of all the batches as the majority of sample point lie on the glacier. A consistent linear trend can be discerned in all

the sample points, whether it be acceleration (upward sloping lines) or deceleration (downward sloping lines). Five noticeable lines do not conform to the overall trend and they are reflective of positive correlations of pieces of ice/debris no longer attached to the glacier following calving events. The exception being sample point 15 where a continuous correlation failed after image 7. Some lines also end abruptly and they indicate that the features at those sample points can no longer be tracked due to their gradual disappearance.

BATCH ONE: 5th of February 2013 *to* 20th of May 2013 (~ 3.5 months or 105 days)

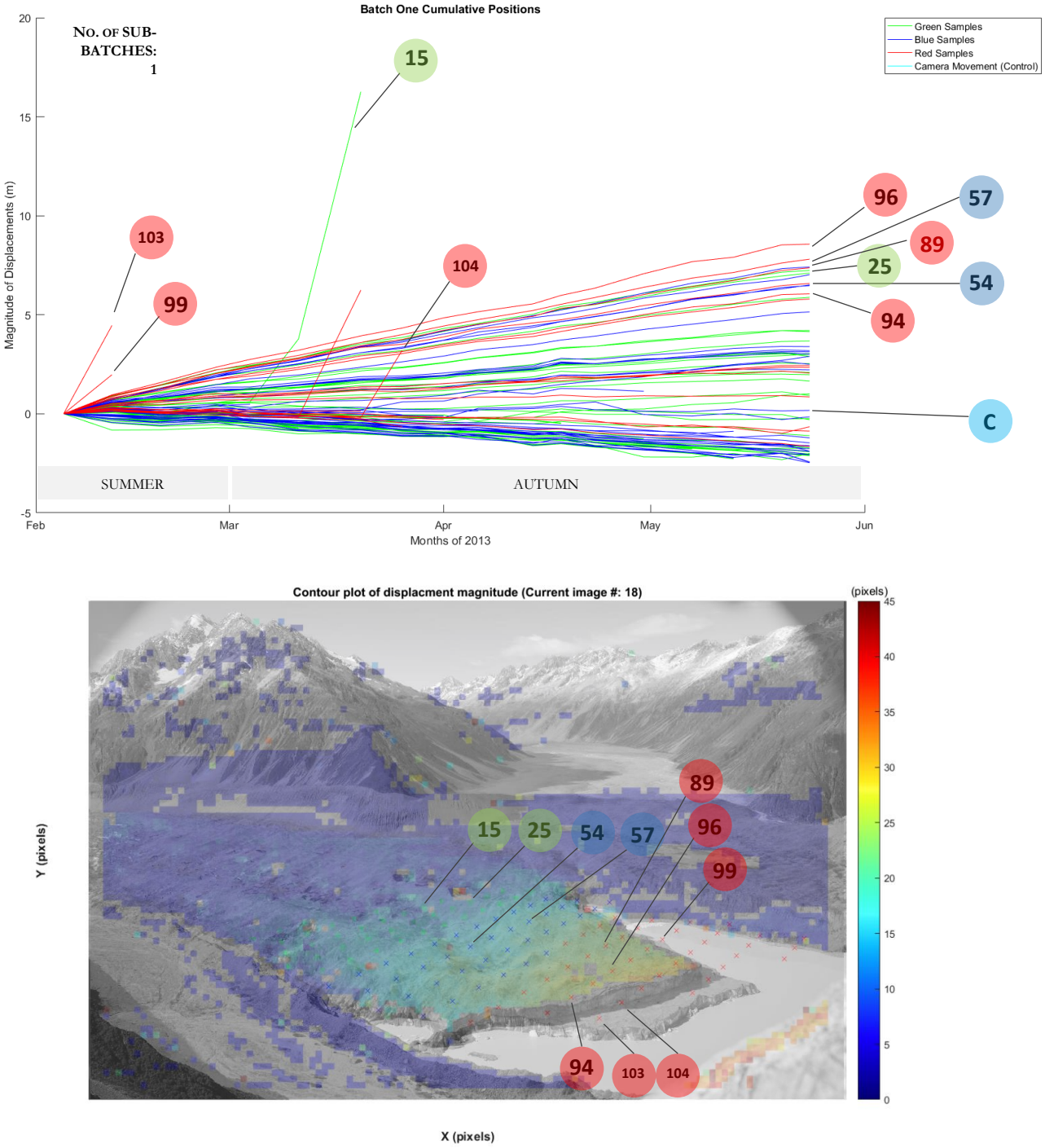


Figure A.1.3.1. (TOP) Shows final displacements in meters for batch one. This is the output after the pixel movements have been corrected for parallax to represent the real-world positions. The slope of the lines indicate the velocity. (BELOW) The net image correlation result for the Batch prior to geo-rectification overlay by the location of the sample points with the first image. **Number of sub-batches is one.** Positions of sample points are for cross-reference with *Figure A.1.3.1. (TOP).*

This spectrum of lines is of great importance, showing how ice flow at the Tasman terminus is inhomogeneous and spatially variable. Several of the sample points are in the negative direction implying that ice is displacing upstream (backwards), even though the majority of lines show a positive displacement (downstream). From analysis of the raw data it appears correlation between images in these upstream displacing regions are inconsistent (i.e. displacement gaps between images, causing the apparent stability). Further study is required to confirm these trends. For this period, the highest positive net displacement (top line in red) was 7.81 m from sample point 96 and the greatest net displacement in the negative direction was - 2.32 m from sample point 17. The maximum velocity for this period was 0.074 m d^{-1} or 27 m a^{-1}.

Glacial positions also varied with time, with some sample points showing consistent negative movement speckled by episodes of positive movement. This variability with time was more evident at the sample points that had a negative slope whereas the sample points that showed a net positive displacement were consistently advancing. This dynamic behaviour could be representative of fracture and deformation in the underlying ice beneath the debris layers. Eighteen images (sub-weekly samples) were used for this ~ 3.5 month period.

A.1.3.2. Batch Two Results

BATCH TWO: 22nd of July 2013 *to* 20th November 2013 (~ 4 months or 121 days)

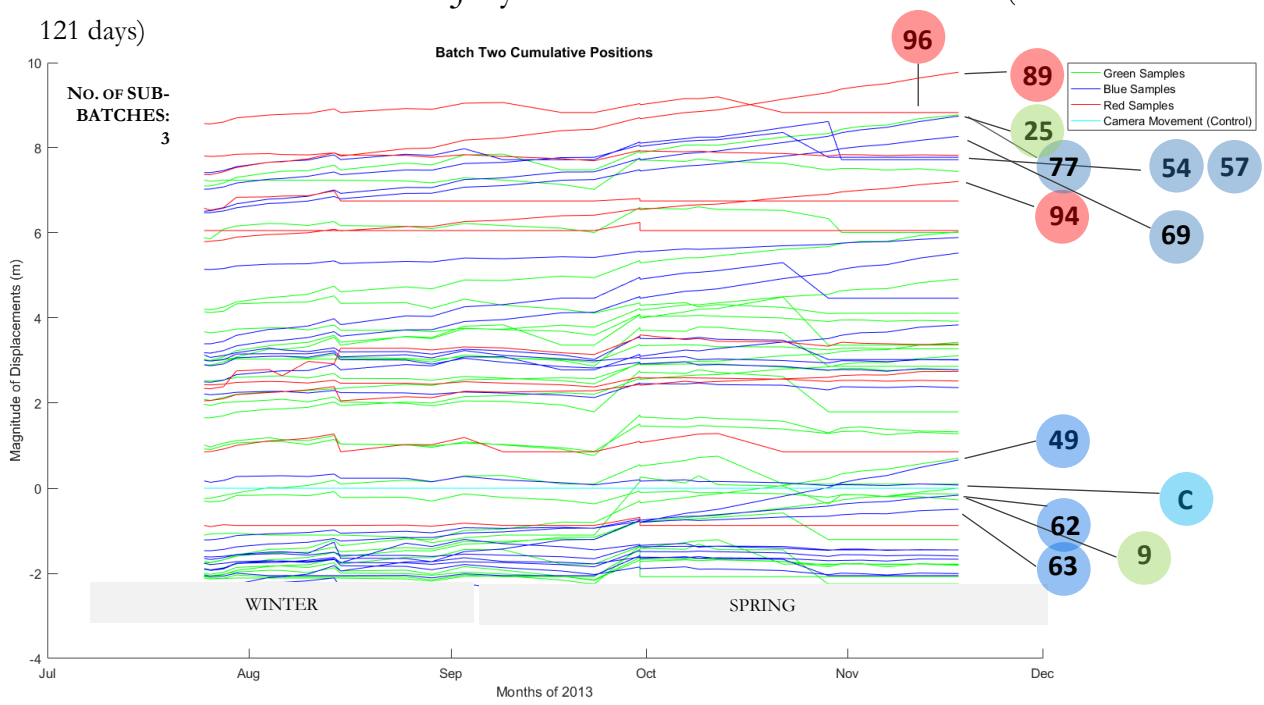


Figure A.1.3.2a. Shows final displacements in meters for batch two. Relatively lower velocity/position changes are apparent during the winter months compared those in spring. C = control and the numbers represent sample points.

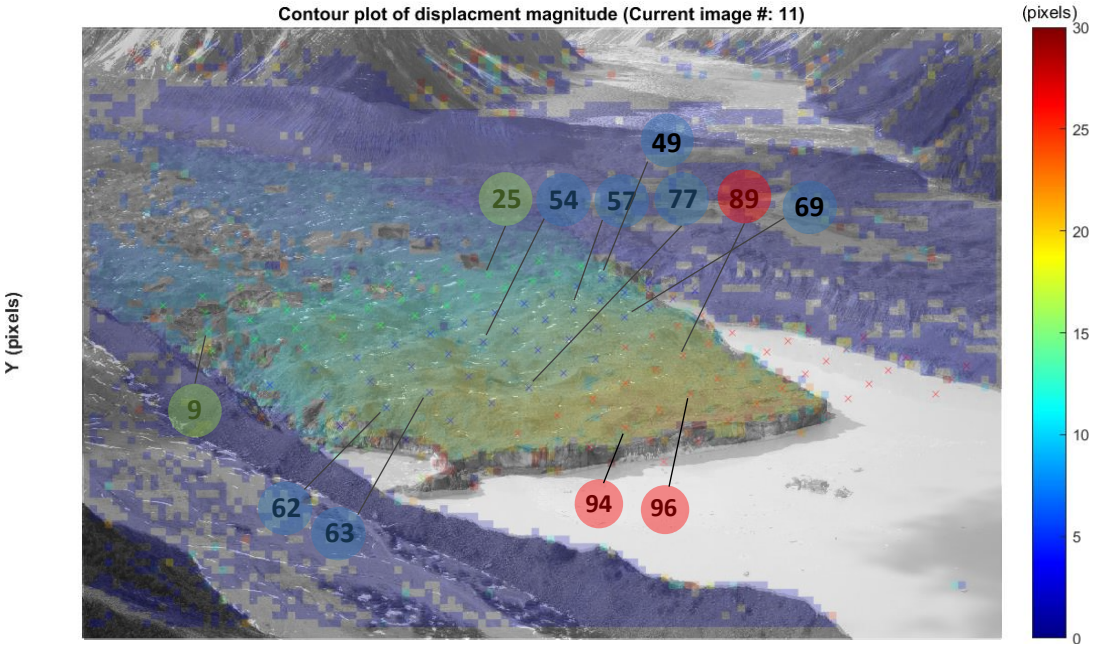


Figure A.1.3.2b. The last image correlation result for the final sub-batch in batch two prior to geo-rectification. Position data from **3 sub-batches** were merged to form Batch Two results. Positions of sample points are for cross-reference with *Figure 4.1.4.2a.*

Following a camera change in July 2013, the frequency of image samples was improved. batch two used 28 images for a period spanning only 4 months (~ average days between images was 4.28). The majority of sample points remained relatively stable and saw significantly less position variability as compared to the first 3.5 months (batch one). An image gap does exist in the first two months of winter so whether this trend persisted through the entire winter of 2013 is inconclusive.

Overall, the maximum displacement among all sample points is 2.41 m (sample point 89) and the minimum displacement being – 0.15 m (sample point 3). Thus the maximum velocity was 7.27 m y^{-1} or 0.0199 m d^{-1} for this period. This is almost a quarter of the maximum velocity in batch one.

Upon closer inspection of the results, temporal complexities exist with some sample points which had previously experienced a negative movement in batch one, displacing forward by a comparable magnitude in the winter and spring months of batch two (i.e. at sample point 9). Another example is seen in sample 49 on the furthest margin near the intersection of the Murchison and Tasman glaciers. In batch one, it had moved - 2.4 m, only to advance ~ 3 m in batch two to a cumulative position of 0.6 m relative to the first image. There are significant imhomogeneities on spatial and temporal levels. Therefore further exploration of these variabilities will be presented in the discussion chapter.

A.1.3.3. Batch Three Results

BATCH THREE:

6th of December 2013 *to* 14th February 2014 (~ 71 days)

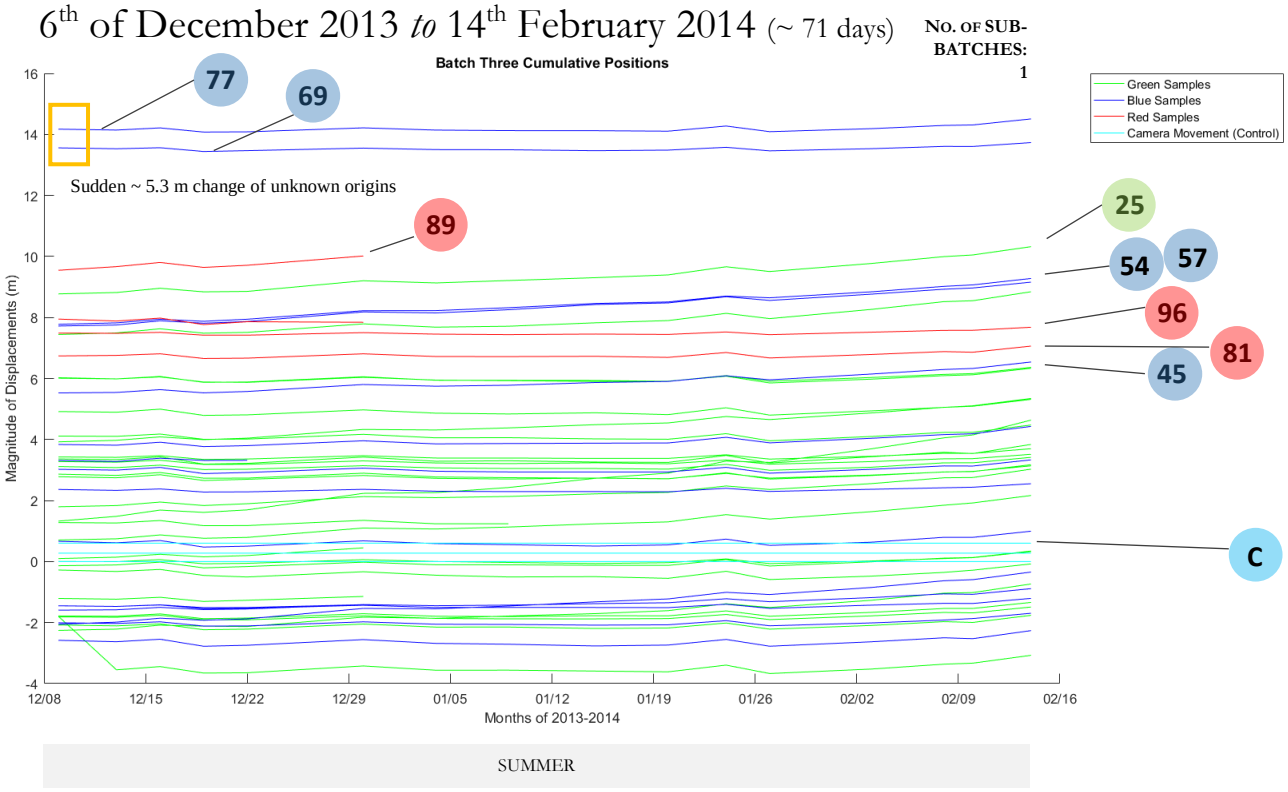


Figure A.1.3.3a. Shows final displacements in meters for Batch Three. C = control and the numbers represent sample points.

The first two months summer of 2013-2014 experienced a consistent linear-like trend in the position data analogous to the winter and spring of 2013. Several sample points failed to be tracked due spatial variabilities from image to image (and are represented by early terminating lines e.g. in SP89). Two sample points (69 and 77) experienced an abrupt ~ 5.3 m advance (SP69 by 5.43 m and SP77 by 5.29 m). Adjacent sample points (spaced 100 m apart) did not provide further evidence of this change by failing to produce a magnitude result (i.e. an output of NaN). A finer grid resolution (20 m or 50 m gridded samples) in a future study may provide further insight into this anomaly.

This batch is similar to batch one in that only one sub-batch was used to attain pixel correlation data. All other Batches (2, 4 and 5) had multiple sub-batches which were assimilated for presentation in the results. Thus, *Figure 4.1.4.3b* represents the net correlation output in the image space for this period. The pixel movements cannot be construed literally however because of parallax errors associated with the camera orientation.

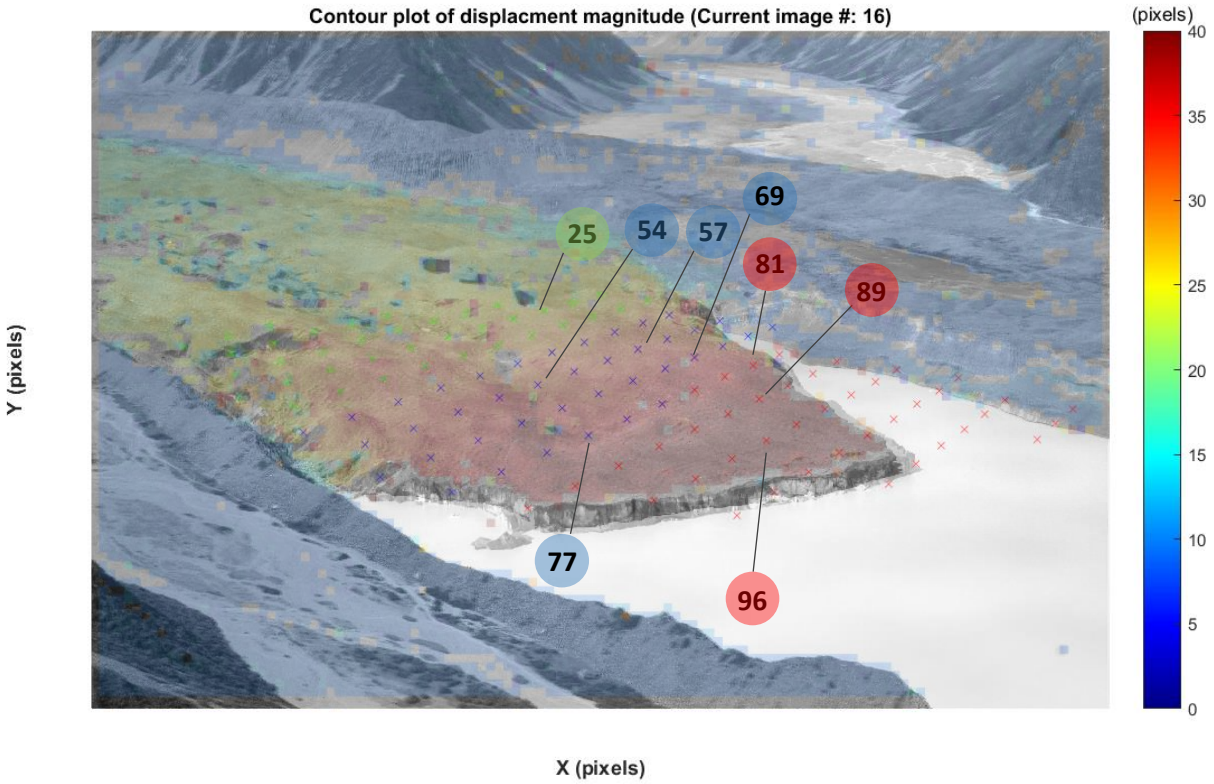


Figure A.1.3.3b. The last image correlation result for the final sub-batch in batch three prior to geo-rectification. Position data from **1 sub-batch** is conveyed in this result. Positions of sample points are for cross-reference with *Figure 4.1.4.3a.*

The DIC algorithm failed to find correlations for some regions of the terminus. One example is in sample point 90 which had previously experienced the highest velocity of all samples in 2013, and was no longer tracked possibly due to calving on the Eastern Embayment Ice Cliff (EIC) of which it is near. Only 2 of the initial 40 red samples are producing a cumulative correlation result despite 14 red samples still situated on the debris surface of the glacier.

The maximum displacement change for this batch was from sample point 25 which moved 1.5 m or an equivalent velocity of 0.02 m d^{-1} or 7.9 m a^{-1} (annual equivalent). This change was mirrored by sample points 54 and 57 also which moved 1.45 m in the same timeframe.

A.1.3.4. Batch Four Results

BATCH FOUR:

21st of February 2013 *to* 27th August 2015 (~1 year and 6 months or 553 days)

[**491 days** of observations]

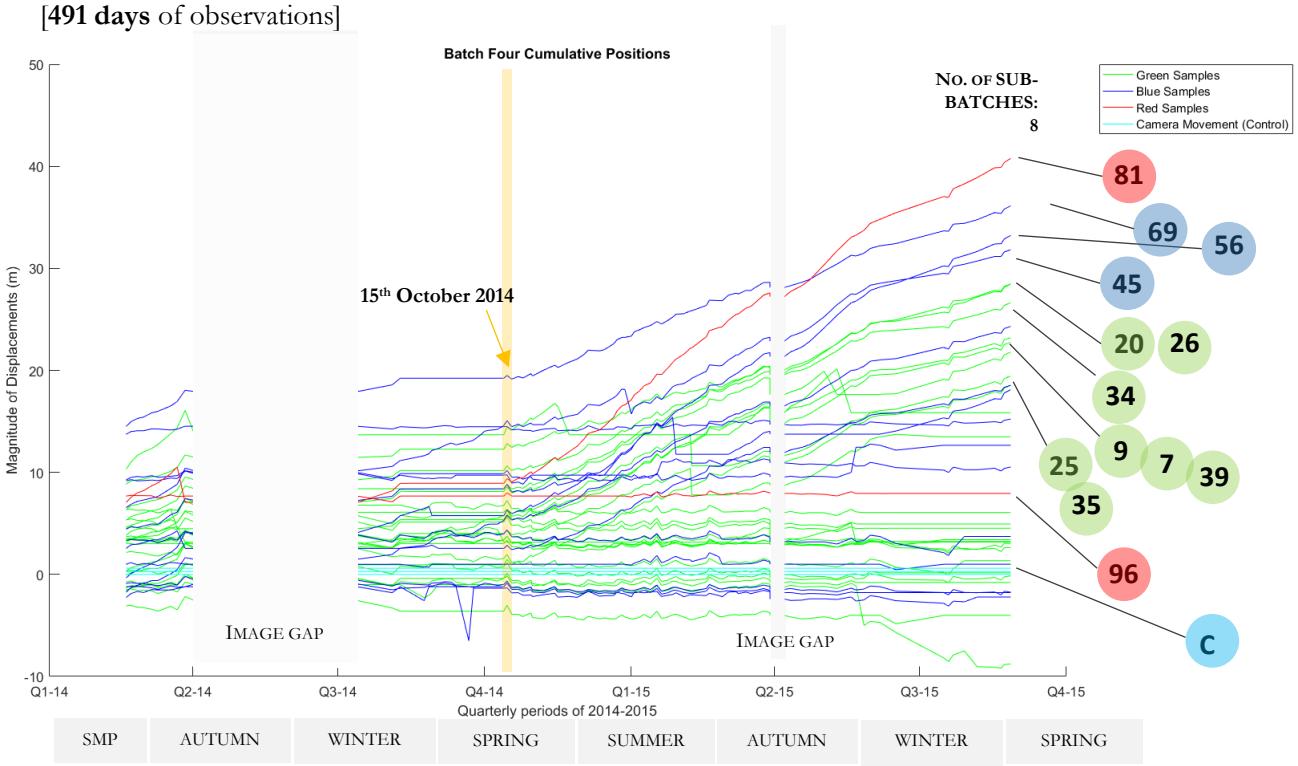


Figure A.1.3.4a. Shows final displacements in meters for batch four. C = control and the numbers represent sample points.

Cumulative results from batch four exhibits merged outputs from 8 sub-batches and contains two periods of non-data. For the non-data periods, cumulative displacements are kept unchanged from the last time point till data is available again. This implies that the net annual displacement is incomplete temporally however spatial, seasonal and inter-annual comparisons can still be attained. Spatial variations in velocity are most stark from batch four results with a handful of sample points departing from a stable trend post summer of 2014-2015. The stable trend is attributed to null correlations occurring within the sample points.

The maximum displacement was recorded from sample point 81 (red) which displaced a total of 33.71 m from 7.06 m in Feb 2014 to 40.77 m in August 2015.

Similar increase in velocities over 2014-2015 were seen in 16 of the samples, led by sample 81 and followed by several from the blue and green categories. Six blue (inc. SP56, SP69, SP45) and 8 green samples that followed this acceleration were previously quite stable in batches 2, 3 and even the first year of batch 4. Sample 25 which was identified as the fastest moving in batches 2 and 3, actually experienced deceleration before recovering to a

cumulative magnitude of 15.85 m. In comparison, samples 26, 20, 34, 9, 7 trumped sample 25 in 2014-2015 and displaced to ~ 28.46 m, 28.44 m, 26.63 m, 23.19 m and 22.72 m respectively. Meanwhile, blue samples 45 and 69 stayed at the top of the blue category although SP56 experienced a much greater velocity of 31.1 m a^{-1} compared to 19.3 m $^{a-1}$ and 19.1 m a^{-1} in the former over the same period.

Of particular note is sample point 96 which plateued throughout 2014-2015. Upon inspection of the timeseries, roughly 30% of the data failed to correlate (e.g. stored as NaN). However, for the timesteps that did correlate, no significant changes were detected that coincided with the acceleration experienced by other samples (i.e. SP81). Due to the geometry of the camera, it may appear as though changes are more significant at the terminus yet after conversion to the real-world displacements, such changes are less apparent as evidenced in SP96.

The maximum annual annual velocity (SP81) was 36.26 m a^{-1} (equivalent) compared to 3.66 m a^{-1} prior to the speed up on the 15th October 2014. This annually averaged velocity is the second highest of all batches , behind batch five. **N.B.** Velocities are calculated by dividing the net displacement by the number of days of observations. SP81 averaged entire the entirety of batch four was 25.06 m a^{-1}. This will exclude the duration where there is an image gap.

A summary of each sample point's velocity can be found in *4.1.5.*

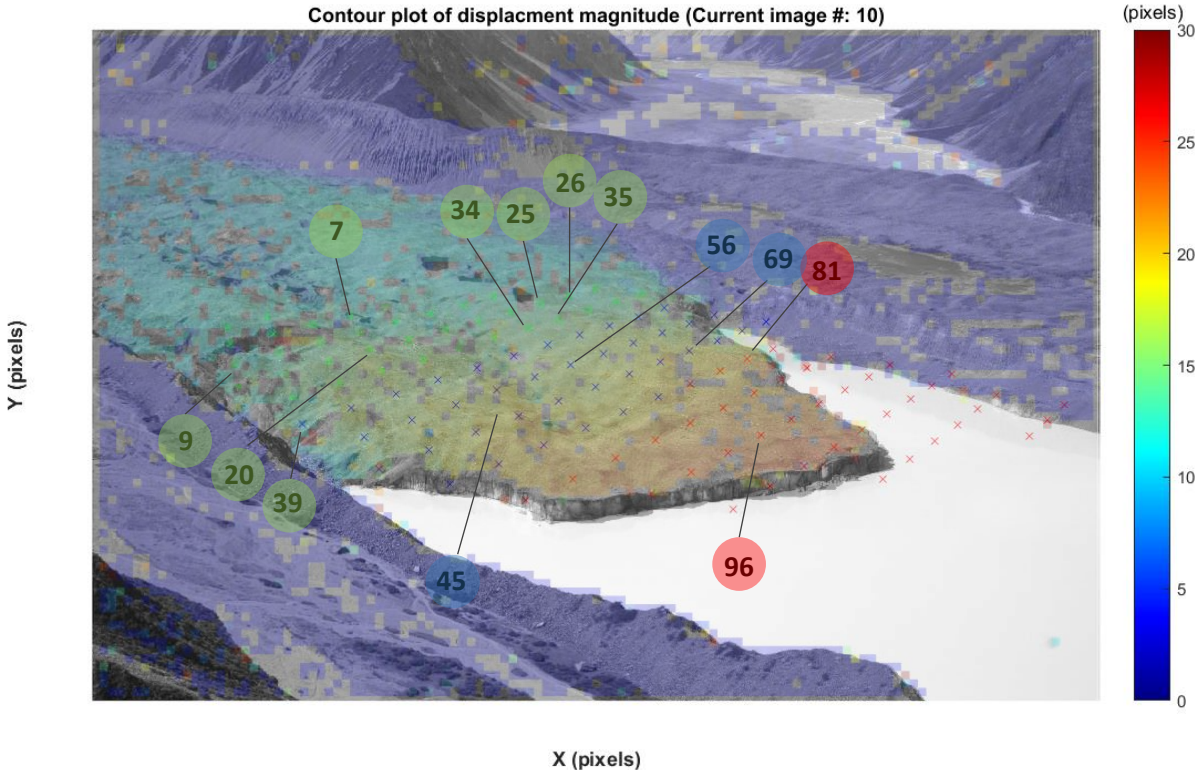


Figure A.1.3.4b. The last image correlation result for the first sub-batch in batch four prior to geo-rectification. Position data from **8 sub-batches** is conveyed in *Figure 4.1.4.4a.* Positions of sample points shown in this figure are for cross-reference with *Figure 4.1.4.4a.*

A.1.3.5. Batch Five Results

BATCH FIVE:

27th of August 2015 *to* 22nd March 2016 (~ 7 months or 209 days) [**108 days** of observations]

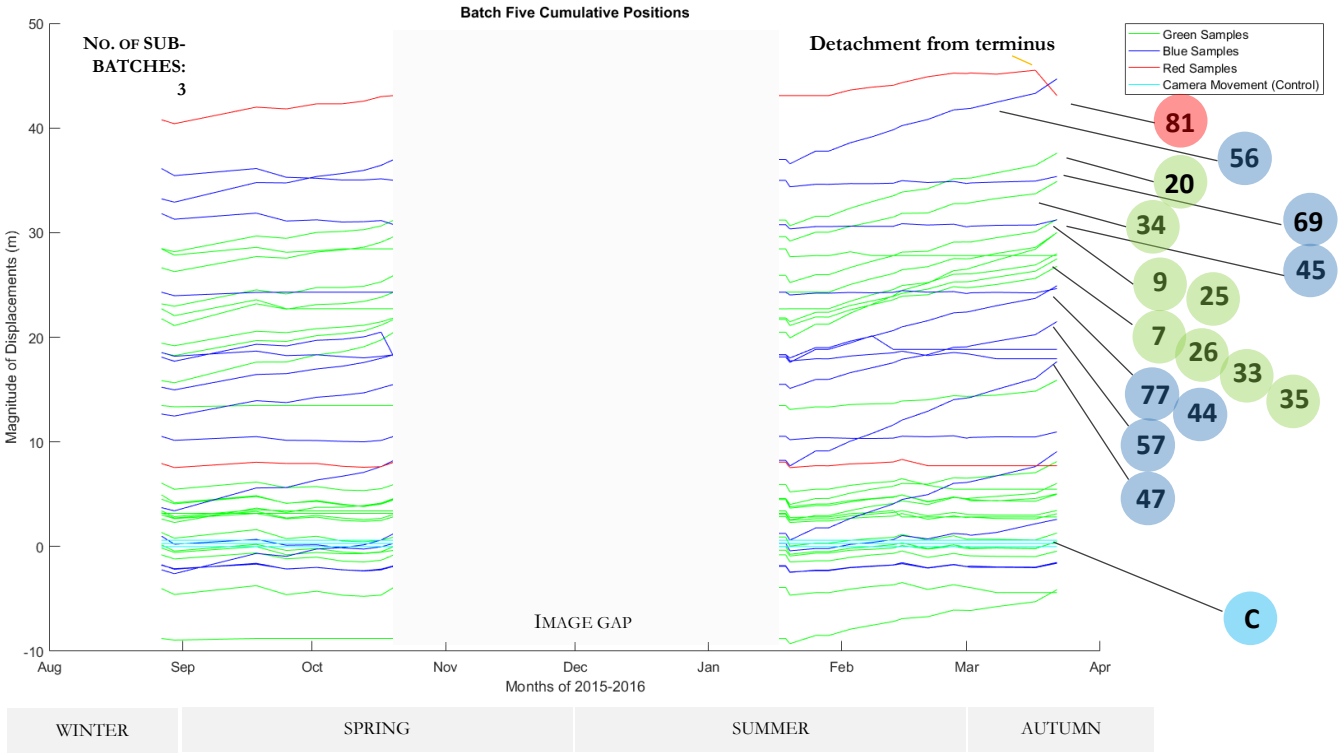


Figure A.1.3.5a. Shows final displacements in meters for Batch Four. C = control and the numbers represent sample points.

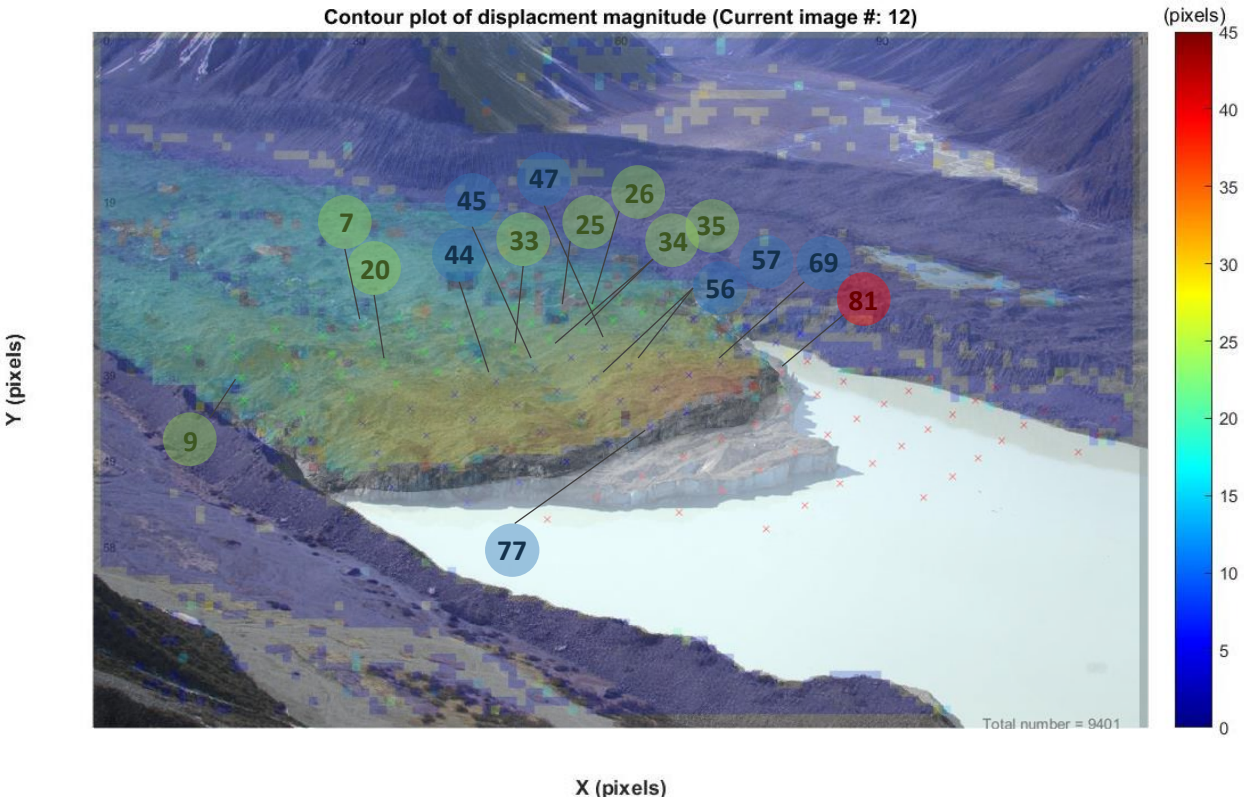


Figure A.1.3.5b. The last image correlation result for the last sub-batch in batch five prior to geo-rectification overlaid by the first image of the batch (before the two calving events of 2016). Position data from **3 sub-batches** is conveyed in *Figure 4.1.4.5a.* Positions of sample points shown in this figure are for cross-reference with *Figure 4.1.4.5a.*

The final batch showed a continued displacement trend in SP81 and numerous sample points from the blue and green category. SP81 ends with a sudden drop in position, which may be attributed to a false correlation as the terminus retreats behind SP81 which meant that it could no longer be tracked effectively. This is the second instance where SP81 experienced a drop in cumulative position in the entire time series (a depreciation also occurred in batch four) and based on review of the images, we should expect a cease in position data from SP81 post-22nd March 2016.

The greatest magnitude of position change occurred in SP25 (green) from 15.65 m in August 2015 to 30.01 m in March 2016. The velocity is calculated by dividing the net displacement by the number of days of observations which was 108 days. The time-averaged velocity for this batch was therefore 0.13 m d^{-1} or an equivalent annual rate of 48.53 m a^{-1}. This significant velocity was also mirrored in SP47 (blue) with 48.16 m a^{-1}.

A.1.4. Calculating velocities

The end positions of the following samples points were selected to calculate the flow rate/velocity of the glacier in *Table 4.1.5a*. Sample points that were not chosen had two characteristics:

(1) They appeared to remain unchanged in the cumulative plots because they had incorrect cumulative positions due to a failure to correlate some images in the time series. When these images did not correlate well, the DIC algorithm allocated a NaN to its value. When the positions were totalled, they ended up not changing its in overall position even though some correlations were made (which explains the oscillations seen in each of those sample points).

(2) They no longer represented changes on the glacier as the sample point no longer lay over the terminus. The majority of these fell into the red category which were the 39 sample points created < 450 m from the ice cliff in February 2013.

Sample points that were chosen for this further analysis consisted of those that demonstrated a continued time series across all five batches and those that did not present any null correlations (i.e. no NaNs across all time steps) throughout the three year study period.

Figure A.1.4a. Presents the 20 chosen sample points relative to the initial 100 m grid. These sample points were plotted as a spatial plot of ice flow using interpolation techniques in *A.1.4.1. Rear image source:* LANDSAT 30 m captured on 24th April 2013

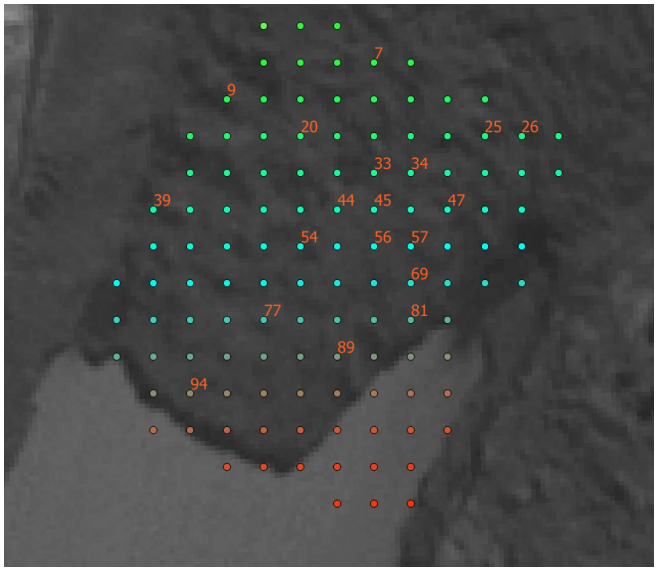


Table A.1.4a. Shows the **final positions** in each batch from time-continuous sample points that did not have any correlation gaps. The exception being SP94 and SP90 which could not be detected post-batch two as they could no longer be tracked.

Sample Point Number	Colour	Start Position (5th February 2013)	Batch 1 Position (m)	Batch 2 Position (m)	Batch 3 Position (m)	Batch 4 Position (m)	Batch 5 Position (m) (22nd March 2013)
94		0	5.80	7.21	N/A	N/A	N/A
89		0	8.56	8.82	N/A	N/A	N/A
81		0	2.34	3.38	7.06	40.77	43.09
77		0	6.47	8.26	13.73	14.97	24.91
69		0	6.52	8.74	14.51	35.44	35.36
57		0	7.41	7.72	9.16	12.47	21.49
56		0	-2.06	-2.05	-0.34	32.91	44.68
54		0	7.03	7.78	9.28	10.15	10.95
47		0	3.07	3.02	3.33	3.41	17.66
45		0	3.17	5.52	6.54	31.28	31.21
44		0	-1.92	-2.00	-1.69	23.97	24.65
39		0	2.89	3.36	3.84	21.12	30.00
35		0	-2.09	-2.08	-1.76	18.25	26.78
34		0	5.85	6.00	6.33	26.27	34.89
33		0	2.52	2.78	3.13	19.18	27.49
26		0	7.24	7.44	8.84	27.85	27.80
25		0	7.10	8.77	10.32	15.65	30.01
20		0	3.65	3.92	5.31	28.17	37.59
9		0	-1.07	-0.28	-0.07	22.96	31.27

Table A.1.4b. Shows the **ice velocities (m a^{-1})** in each batch calculated from time-continuous sample points that did not have any correlation gaps in *Table A.1.4a*. The exception being SP94 and SP90 which could not be detected post-batch two as they could no longer be tracked. Annually averaged velocities are calculated by first calculating the displacement of each batch then divided by the number of days in the respective batch. This gives the daily averaged velocity, which is then multiplied by 365.25 days for equivalent annually averaged rates (m a^{-1}.e). The highest two velocities in each batch are highlighted in bold.

No. Days			105	121	71	491	108
Sample Point Number	**Colour**	**Start Position** (5th February 2013)	**Batch 1 Velocity (m a^{-1}.e)**	**Batch 2 Velocity (m a^{-1}.e)**	**Batch 3 Velocity (m a^{-1}.e)**	**Batch 4 Velocity (m a^{-1}.e)**	**Batch 5 Velocity (m a^{-1}.e)**
94		0	20.16	4.25	N/A	N/A	N/A
89		0	**29.76**	0.78	N/A	N/A	N/A
81		0	8.13	3.14	18.92	**25.06**	7.84
77		0	22.49	5.40	**28.12**	0.92	33.59
69		0	22.66	**6.70**	**29.66**	15.56	-0.27
57		0	**25.76**	0.94	7.40	2.46	30.48
56		0	-7.16	0.03	8.79	24.72	39.78
54		0	24.44	2.26	7.71	0.65	2.70
47		0	10.67	-0.15	1.59	0.06	**48.16**
45		0	11.02	**7.09**	5.24	18.39	-0.24
44		0	-6.67	-0.24	1.59	**19.08**	2.30
39		0	10.05	1.42	2.47	12.85	30.01
35		0	-7.27	0.03	1.65	14.88	28.83
34		0	20.34	0.45	1.70	14.82	29.13
33		0	8.76	0.78	1.80	11.93	28.08
26		0	25.17	0.60	7.20	14.13	-0.17
25		0	24.68	5.04	7.97	3.96	**48.53**
20		0	12.69	0.81	7.15	16.99	31.84
9		0	-3.72	2.38	1.08	17.12	28.08
7		0	14.36	-0.06	1.85	13.08	20.01
Mean		**0**	**13.32**	**2.08**	**7.88**	**12.59**	**22.71**

From *Table A.1.4a*, it is clear that position changes in various sample points across the glacier are highly dynamic. Speed up are also occurring on different spatial scales. For example, SP44 and SP39 showed relatively slow movement in batches 1-3 (2013-2014) but experienced significantly higher movement in batch 4 and 5 (2014-2016). It is not clear what underlying processes may be governing the spatial and temporal diversity of these

observations. *Table 4.1.5b* summarises the equivalent velocities from these sample points if they were projected to annual rates.

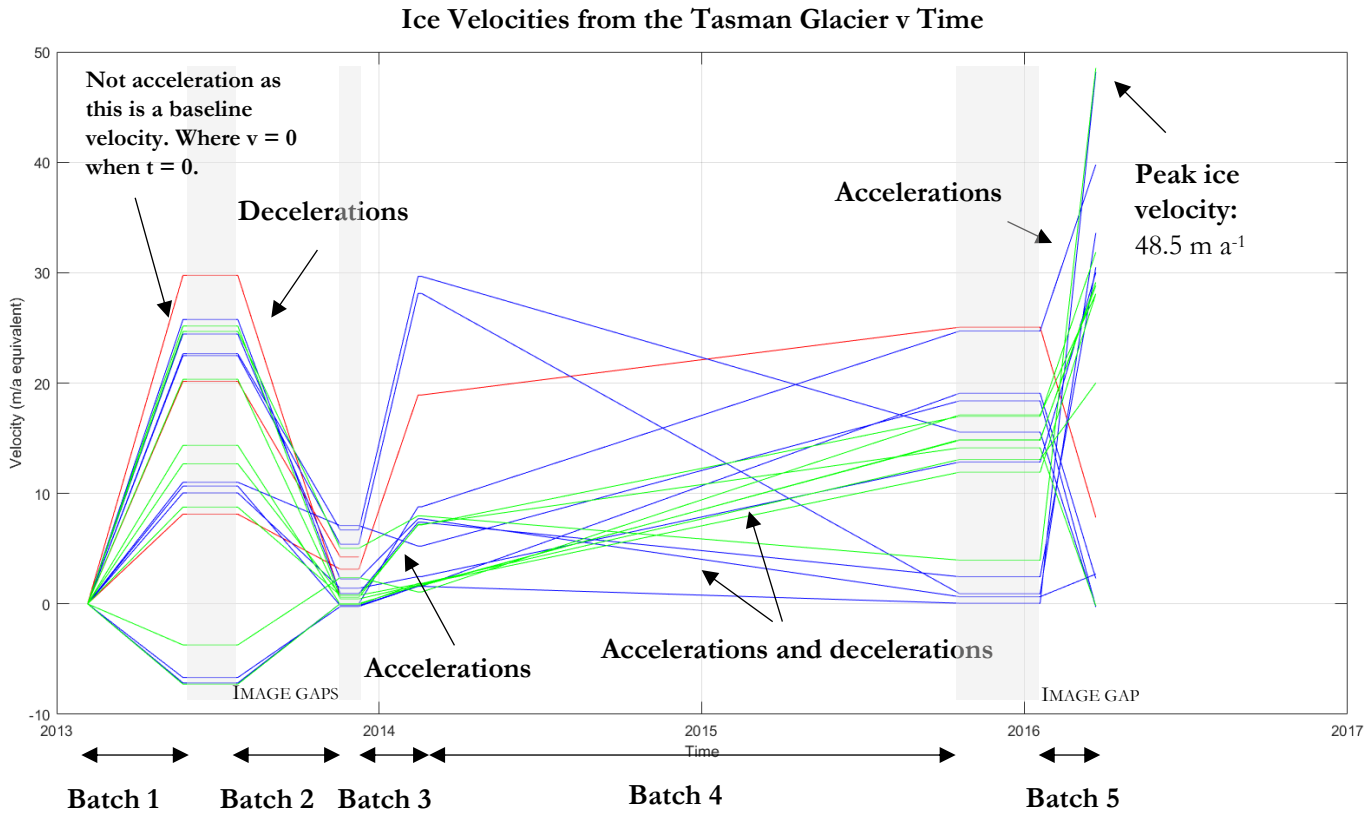
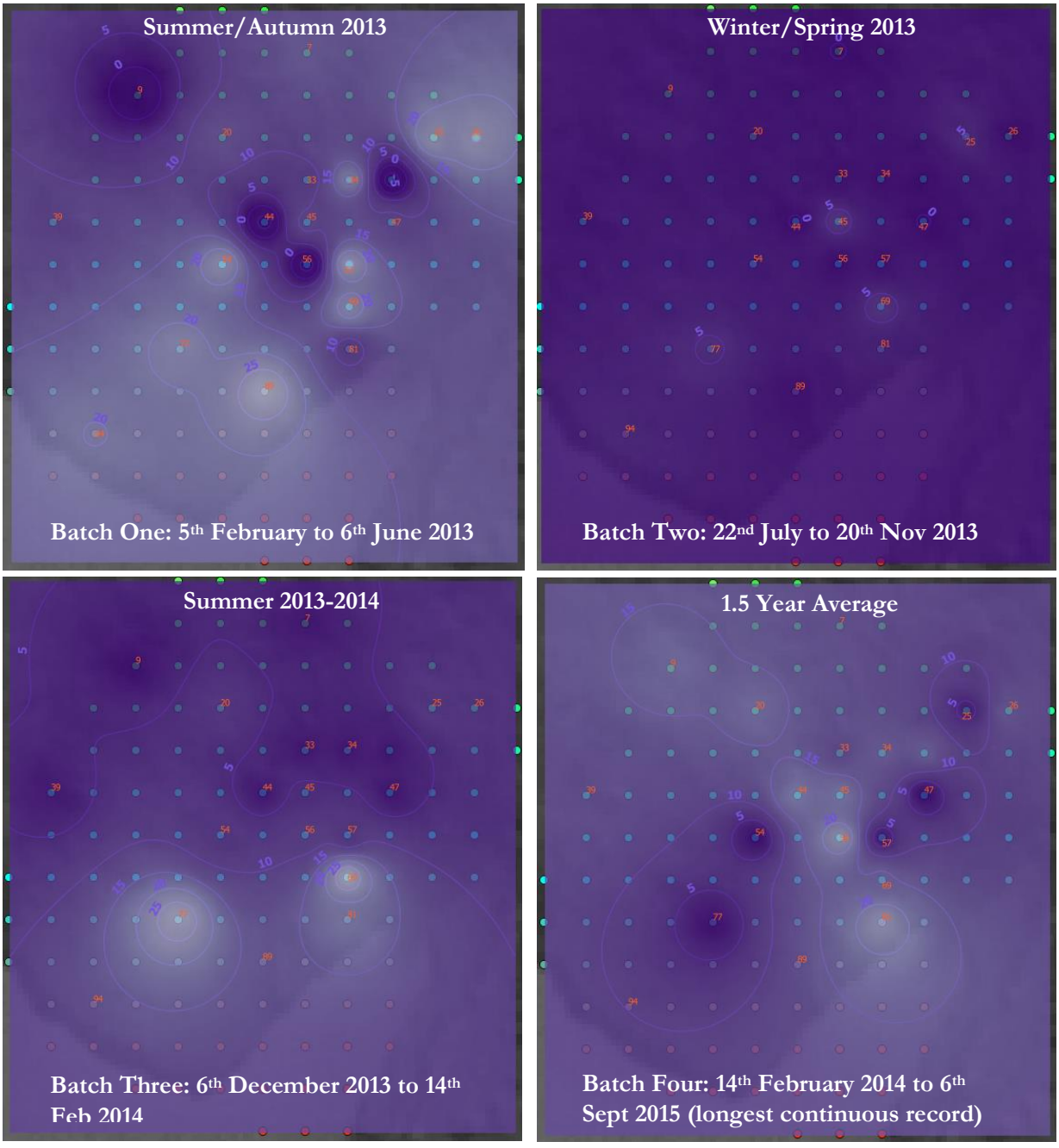


Figure A.1.4b. Shows the **ice velocities (m a^{-1}** equivalent**) plotted vs time.** Velocities were calculated using batch displacements rather than discretised displacements at each time step. This gives a good general depiction of changes in velocity over the study period. A more detailed study in the future can encompass discretised velocities in order to capture velocity changes over smaller timescales.

The 20 sample points from *Table 4.1.5b* were plotted to depict temporal patterns via accelerations and decelerations. The velocities from the first batch serve as a baseline for comparison as velocities are to be zero from day one. *Figure 4.1.5* shows ice velocities varying significantly on temporal scales. A noticeable deceleration occurred in the majority of sample points from the winter and spring months of batch two (22nd July to 20th November 2013). Ice velocities accelerated again in the summer of 2014 during batch three (6th December 2013 to 14th February 2014). In batch four, some of the sample points (SP77, SP54 and SP47) that exhibited the highest velocities in batch three stagnated and did not change in position by much from 14th February 2014 to 6th September 2015. Meanwhile, some sample points continued to accelerate through batch four. In the summer of 2016 (batch five), the highest recorded ice velocity was observed in SP25 (48.5 m a^{-1}) and was closely followed by SP47 at 48.1 m a^{-1}.

These results indicate that velocities are highly variable and possibly oscillatory. A seasonality to the accelerations and decelerations may also be present. The volatility of the ice flow over temporal scales on the Tasman Glacier is best epitomised by the divergence and intersection of velocity lines. This temporal dynamism is also represented by the partly cyclical phases of acceleration and deceleration experienced at every sample point. Moreover, the separation of velocity lines indicates a spatial inhomogeneity in ice flow although the differences in velocity between sample points are less prominent in some regions. An exploration of the spatial distribution of velocity is presented the next sub-chapter.

A.1.4.1. Spatial distribution of ice flow: Results



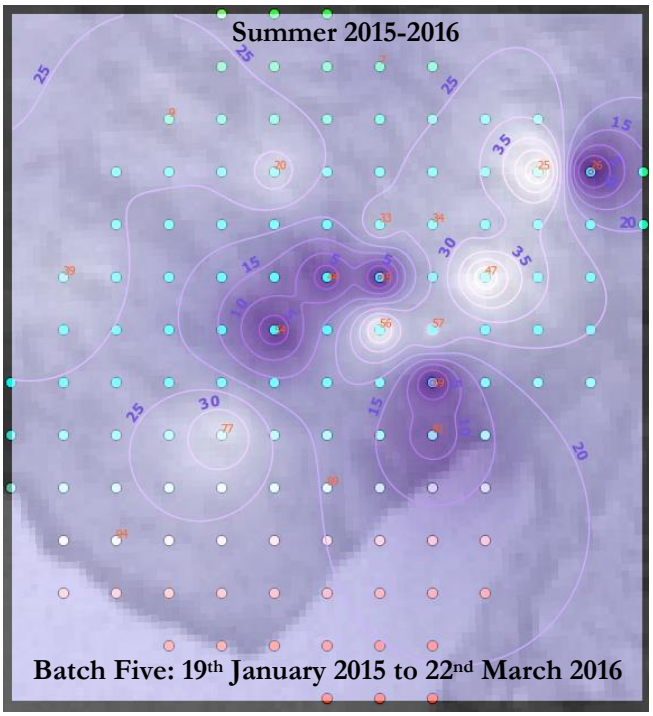


Figure A.1.4.1. Shows the **ice velocities (m a^{-1}** equivalent**)** presented in QGIS using Inverse-Distance Weighting (IDW) interpolation of the 20 sample points in *Figure A.1.4*. Contours are spaced evenly at 5 m a^{-1} and highlighted in orange are the respective real world positions of the 20 sample points. No changes were made to the image saturation and brightness after interpolation. Only the transparency of each result was altered post-interp. They were all consistently changed to 25%.

As reference, the brighter the interpolation, the faster the ice flow. Dark regions were regions of 0 or negative velocities. A fixed satellite image is presented in all batches for comparisons. *Base Image: LANDSAT (30 m)* 24th April 2013.

The spatial patterns observed in the first year of observations shows a stark difference in ice velocities between seasons. During the end-of-summer/autumn months of 2013, ice velocities were as high as 29 m a^{-1} e. (SP89) compared to a maximum observed ice flow of 7.09 m a^{-1} (SP45) in the winter/spring months of the same year. Comparisons between batches two and three also reinforce the observations with changes at SP77 and SP69 reflecting a similar speed up during summer months. It is not fully clear what driving mechanisms are affecting these changes in the ice.

Batch four, which represents the longest uninterrupted record elapsing 1.5 years showed an annually-averaged distribution of ice flow at the Tasman for 2014-2015. SP69 exhibited high velocities along with the nearby SP81. Interestingly, SP77 actually decelerated over 2014-2015. But accelerated back to 33.59 m a^{-1} in the summer of 2015-2016.

Overall, the greatest ice velocities are seen in batch five peaking in SP47 and SP56. This would have likely affected the coloration of batch five significantly, which explains why it is visibly lighter than all the other batches. Further investigation into the seasonality of ice flow is warranted from the evidence provided in this study.

PlotHistograms.m

This small segment of code was simply used to plot the differences between a good image histogram and a bad image histogram, which would be used to find the optimal images for processing. This falls under the Image Selection Phase.

```
%%%% SHOW HISTOGRAM OF BAD IMAGE and GOOD IMAGE %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

A = imread('20130825140033.jpg');
B = imread('20130822110033.jpg');
Agray = rgb2gray(A);
Bgray = rgb2gray(B);

figure(1)
[pixelCounts, grayLevels] = imhist(Agray,256); % Specify number of bins
bar(grayLevels, pixelCounts, 'FaceColor', [0.19 0.99 0.19]);
ylabel(['Pixel Count'])
xlabel(['Gray Levels (0 - 256)'])
legend(['Good image'])
figure(2)
[pixelCounts, grayLevels] = imhist(Bgray,256)
bar(grayLevels, pixelCounts, 'FaceColor', [0.93 0.125 0.125]);
ylabel(['Pixel Count'])
xlabel(['Gray Levels (0 - 256)'])
legend(['Bad image'])
```

Automate_Transect_Intersections.m

This code was used to calculate the relative position of vectorised Terminus Positions for part two of my results which presented the annually averaged Rates of Retreat (U_r) and Surface Area Flux (dA/dt) for 2013-2016. The Surface Area Flux code was introduced at the end of this code. The majority of the code focusses on importing and processing matrices of positions from the Vectorised Termini and the projected Transects (1-20). The Transects had to be digitally generated with the same slope. Additionally, because of the nature of the image samples, four batches meant every step of the process had to be replicated four times. There was not a simpler method due to the variations in image resolution, image quantity and vector length (from the manual digitisation – which did constrain the number of points that formed a terminus vector) for each Batch.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ---------------------------------------------------------
% AUTOMATE_Transects_Intersections Script (2016) - Lui,E
% ---------------------------------------------------------
% Automating the transect intersections with a line drawn on QGIS
% First undistort the shapefiles
% Then with the undistorted shapefiles, draw a transect in QGIS. Save that
% shapefile and import into matlab
% Then find the intersections to PRECISION and save it as a file
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% LOAD 1.0485 Slope File
load('C:\Users\Edmond\Desktop\THE THESIS\[10] OUTPUTS\1.4085
slope\AllDataFiles.mat')

% Add path to m_map tools
clear all
clc
addpath /Users/Edmond/Desktop/m_map
addpath /Users/Edmond/Desktop/THE THESIS/[6] Measuring the displacement from
Transects/ % Intersection.m script
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% (1) Read in Shapefiles, Must be shapefiles not geojson. Hint- Use ogr2ogr
% on QGIS to convert to the correct shapefile - make sure to save it as
% Polyline with No Z elevation

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% (2) Load in ShapeFiles ['Lines', 'Attributes'] Format

[FirstBatch, Date_FirstBatch] = shaperead('UNPROJECT_6months_2d.shp'); % Matlab
Cannot read PolyLinZ because that is a 3d line with elevation. First convert it to
a 2d space.
[Pre20thBatch, Date_Pre20th] = shaperead('UNPROJECT_Pre20th_2d.shp');
[Post20thBatch, Date_Post20th] = shaperead('UNPROJECT_Post20th_2d_sort.shp');
[LastBatch, Date_LastBatch] = shaperead('UNPROJECT_LastBatch_2d_sort.shp');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% (3) Plot to see all shapefiles
% Convert from Structure to a Cell for plotting
Date_First = struct2cell(Date_FirstBatch);
Date_Second = struct2cell(Date_Pre20th);
Date_Third = struct2cell(Date_Post20th);
Date_Fourth = struct2cell(Date_LastBatch);

figure(1)
%ax.XLimMode = 'manual'; % Changes Xlimit Mode to Manual rather than Auto
%ax.YLimMode = 'manual';
%xlim([5.1614e6, 5.1623e6])
%ylim([1.3726e6, 1.374e6])
plot([FirstBatch.X], [FirstBatch.Y], 'k')
hold on
plot([Pre20thBatch.X], [Pre20thBatch.Y], 'k')
plot([Post20thBatch.X], [Post20thBatch.Y], 'k')
plot([LastBatch.X], [LastBatch.Y], 'k')
ax.XGrid = 'on'; % Toggle X Grid Lines
ax.YGrid = 'on'; % Y grid
axis manual % tight, fill, etc.
grid on

title('Tasman Glacier Terminus Positions February 2013 - March 2016')
xlabel ('X')
ylabel ('Y')
```

```
% (3a) GUINPUT for Transect Boundary - DRAW THE TRANSECT!
% PLOTTING THE FIRST REFERENCE TRANSECT - Right Hand Side
[Lx,Ly]=ginput(1);
[Lx2, Ly2] = ginput(1);
A1 = [Lx, Lx2]; % Need to be in this format so as not to plot a line that FITS the
points, rather a line that only fits the two points
% Get the Bottom location of transect clicked
A2 = [Ly, Ly2];
scatter (A1, A2);
plot(A1, A2);
X = [Lx, Ly; Lx2, Ly2];
Dist = pdist(X, 'euclidean'); % Calculate the distance between 2 points

Aslope = (Ly2 - Ly) ./ (Lx2 - Lx);
str = num2str(Aslope);
text(Lx, Ly, str);

% Plotting more Transects Parallel to Original One - LEFT HAND LIMIT

[Lx3, Ly3] = ginput(1);
[Lx4, Ly4] = ginput(1);
B1 = [Lx3, Lx4];
B2 = [Ly3, Ly4];
scatter (B1, B2);
plot(B1, B2);
Bslope = (Ly4 - Ly3) ./ (Lx4 - Lx3);
%str = num2str(Bslope); % Converts the scalar to a STRING to be presented
X = [Lx3, Ly3; Lx4, Ly4];
Dist2 = pdist(X, 'euclidean'); % Calculate the distance between 2 points
Dist2
text(Lx3, Ly3, str);

% Automate Generation of Transects via TRANSLATION +/- Translation

Nx = Lx - 100; % Changes based on the translation occuring
Nx2 = Lx2 - 100; % Changes based on the translation occuring
Ny = Ly + 5 ; % Fixing the elevation
Ny2 = Ly2 + 5 ; % No change to elevation 2
Bx = [Nx, Nx2]; % Looping Bx, changes with every loop
By =  [Ny, Ny2];
scatter(Bx, By);
plot(Bx, By);
hold off


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (4b) TEST BATCH for FIRST LINE - Create vectors out of the plotted (SYNTAX is
TransectOneX =
% linspace(X, X, ##) where ## is number of increments. This is mandatory in
% order to calculate the intersection between the transect vector and the
% digitized terminus positions.

% RH Reference
TransectOneX = linspace(Lx, Lx2, 371); % Where 371 is the number of vector points
for FirstBatch(1).X
TransectOneY = linspace(Ly, Ly2, 371);

% PROBLEM - Intersection still uses EVENLY-SPACED Vectors for comparison.
% Therefore  we must find the size of FirstBatch(i).X, or Y. As reference
% to how many Linspaces are created.

% INTERSECTION TEST with [x0,y0,iout,jout] =
% intersections(x1,y1,x2,y2,robust) where robust on is 1
% The X0, Y0 is the coordinate of the intersection. The iout is what i
% want, which is how far along the Transect that the intersection occurs
[x0, y0, iout, jout] = intersections(TransectOneX, TransectOneY, FirstBatch(1).X,
FirstBatch(1).Y, 1) %It works
```

```
% Loop for all in First Batch
Size1 = size(FirstBatch); % Size of the Batch
Size2 = size(Pre20thBatch);
Size3 = size(Post20thBatch);
Size4 = size(LastBatch);

% First Batch intersection with Transect
for i = 1:Size1(1)
    BatchX = FirstBatch(i).X;
    VectorSize = size(BatchX);
    TransectX = linspace(Lx, Lx2, VectorSize(2)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
    TransectY = linspace(Ly, Ly2, VectorSize(2));
    [x0, y0, iout, jout] = intersections(TransectX, TransectY, FirstBatch(i).X,
FirstBatch(i).Y, 1); %It works
    FirstIntX(i) = x0; % Output name
    FirstIntY(i) = y0; % Output name
    FirstIntCross(i) = iout * (Dist/VectorSize(2)); % Output of where it is on the
line . DIST - is the distance of the transect in real world
end
% Pre20th
for i = 1:Size2(1)
    BatchX = Pre20thBatch(i).X;
    VectorSize = size(BatchX);
    TransectX = linspace(Lx, Lx2, VectorSize(2)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
    TransectY = linspace(Ly, Ly2, VectorSize(2));
    [x0, y0, iout, jout] = intersections(TransectX, TransectY, Pre20thBatch(i).X,
Pre20thBatch(i).Y, 1); %It works
    SecondIntX(i) = x0; % Output name
    SecondIntY(i) = y0; % Output name
    SecondIntCross(i) = iout * (Dist/VectorSize(2)); % Output of where it is on the
line
end
% Post20th
for i = 1:Size3(1)
    BatchX = Post20thBatch(i).X;
    VectorSize = size(BatchX);
    TransectX = linspace(Lx, Lx2, VectorSize(2)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
    TransectY = linspace(Ly, Ly2, VectorSize(2));
    [x0, y0, iout, jout] = intersections(TransectX, TransectY, Post20thBatch(i).X,
Post20thBatch(i).Y, 1); %It works
    ThirdIntX(i) = x0; % Output name
    ThirdIntY(i) = y0; % Output name
    ThirdIntCross(i) = iout * (Dist/VectorSize(2)); % Output of where it is on the
line / ADJUSTED For changes in the line distibution
end

% Final Batch
for i = 1:26
    if i == 9
        continue % This will skip i = 9
    end

    BatchX = LastBatch(i).X;
    VectorSize = size(BatchX);
    TransectX = linspace(Lx, Lx2, VectorSize(2)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
    TransectY = linspace(Ly, Ly2, VectorSize(2));
    [x0, y0, iout, jout] = intersections(TransectX, TransectY, LastBatch(i).X,
LastBatch(i).Y, 1); %It works
    FourthIntX(i) = x0; % Output name
    FourthIntY(i) = y0; % Output name
    FourthIntCross(i) = iout* (Dist/VectorSize(2)); % Output of where it is on the
line
end
% Error in entry number 9 for Batch 4 where it does not intercept
```

```
FourthIntCross(9) = NaN;
FourthIntCross = FourthIntCross(3:end); % Remove first two entries

% Testing the theory that LinSpace distributes length based on points
%TransectX = linspace(Lx, Lx2, 100);
%TransectY = linspace(Ly, Ly2, 100);
%TransectX2 = linspace(Lx, Lx2, 100);
%TransectY2 = linspace(Ly, Ly2, 100);


% Corrections for the Intersections because of the change in scale
% resulting from the vectorisation using linspace
% SOLUTION - Correct it by a scalar factor relative to linspace
% TEST

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (4c) Attain Data Times for all Batch Digitizations - NEED THIS FOR PLOTTING
% Every Matlab day is 86400 seconds
Date1 = Date_First(1,:);
Date1 = cellfun(@(x) x(1:8),Date1(cellfun('length',Date1) > 7),'un',0); % Truncate
date1 to no more than 7. NEED TO DO THIS BECAUSE
% ~ datenum in the next stage keeps giving random errors as a result of the
% conversion due to MEMORY LOSS
http://au.mathworks.com/matlabcentral/newsreader/view_thread/310427
for i = 1:length(Date1);
    datecount = Date1(i); % Need to convert the DATE to a STRING first using
datestr
    %OLD COMMAND- Datenm2(i) = datenum(datecount, 'yyyyMMdd')
    Datenm1(i) = datenum(cellfun(@(x) {[x(1:4) '-' x(5:6) '-' x(7:8)]}, Date1(i)),
29); % COPIED FROM ABOVE LINK Convert Datetime to Datenum because errorbar function
doesn't like datetime formats
end
% Datenm1 = datenum(Date1, 'yyyyMMddhhmmss'); THIS STATEMENT PRODUCES THE
% WRONG DATES, you need to loop it to attain each date
Date2 = Date_Second(1,:);
Date2 = cellfun(@(x) x(1:8),Date2(cellfun('length',Date2) > 7),'un',0);
for i = 1:length(Date2);
    datecount = Date2(i); % Need to convert the DATE to a STRING first using
datestr
    Datenm2(i) = datenum(cellfun(@(x) {[x(1:4) '-' x(5:6) '-' x(7:8)]}, Date2(i)),
29); % Convert Datetime to Datenum because errorbar function doesn't like datetime
formats
end

Date3 = Date_Third(1,:); % Dont rerun this because Date3(70) is a wrong date.
Should be 2015 not 2014
Date3 = cellfun(@(x) x(1:8),Date3(cellfun('length',Date3) > 7),'un',0); % Truncate

for i = 1:length(Date3);
    datecount = Date3(i); % Need to convert the DATE to a STRING first using
datestr
    Datenm3(i) = datenum(cellfun(@(x) {[x(1:4) '-' x(5:6) '-' x(7:8)]}, Date3(i)),
29); % Convert Datetime to Datenum because errorbar function doesn't like datetime
formats
end
%% ---------------
% ISSUE WITH DATENM3 overlap on 43, 44
%Date3 = Date3([1:43,45:end])
%% ---------------
Date4 = Date_Fourth(1,:);
Date4 = cellfun(@(x) x(1:8),Date4(cellfun('length',Date4) > 7),'un',0); % Truncate

for i = 1:length(Date4);
    datecount = Date4(i); % Need to convert the DATE to a STRING first using
datestr
```

```
    Datenm4(i) = datenum(cellfun(@(x) {[x(1:4) '-' x(5:6) '-' x(7:8)]}, Date4(i)),
29); % Convert Datetime to Datenum because errorbar function doesn't like datetime
formats
end
Datenm4 = Datenm4(3:end); % Because of overlapping dates, removed first 2 August
entries % DISCUSS THIS in THESIS

AllDates = [Date1, Date2, Date3, Date4];
AllDatenm = [Datenm1, Datenm2, Datenm3, Datenm4]; %converts all dates to Datenum
format
% ISSUE: You cannot plot a datestr against integers. You need to plot with Datenum.
Then use datetick to list it.
AllDatestr = datestr(AllDatenm);
disp(AllDatestr)

% Test individual plots
figure(2)
plot(Datenm1, FirstIntCross,'b')
datetick ('x',  'dd-mmm-yyyy');
figure(3)
plot(Datenm2, SecondIntCross,'b')
datetick ('x',  'dd-mmm-yyyy');
figure(4)
plot(Datenm3, ThirdIntCross,'b')
datetick ('x',  'dd-mmm-yyyy');
figure(5)
plot(Datenm4, FourthIntCross,'b')
datetick ('x',  'dd-mmm-yyyy');

% Draft Plot of Transect RH only(reference)
figure(6)
IntersectionAll = [FirstIntCross, SecondIntCross, ThirdIntCross, FourthIntCross];
plot(AllDatenm, IntersectionAll, 'r')
datetick ('x',  'dd-mmm-yyyy'); % converts the X axis to Dates

% testdate = datenum('20150918110027', 'yyyyMMddhhmmss')
% testdatestr = datestr(testdate)

% ------------------------------------------------------------------------------
-------------
% [0] TRANSECT ITERATIONS - Spacing them to generate 20 transects
% ------------------------------------------------------------------------------
-------------
% Automate Generation of Transects via TRANSLATION +/- Translation
Nx = Lx;
Nx2 = Lx2;
Ny = Ly;
Ny2 = Ly2;
clear Bx
clear By
clear Tx
clear Tx2
clear Ty
clear Ty2

for k = 1:20 % 20 Transects spaced evenly. rather than 100

    Tx(k) = Nx - 20; % Changes based on the translation occuring
    Nx = Tx(k);
    Tx2(k) = Nx2 - 20; % Changes based on the translation occuring
    Nx2 = Tx2(k);
    Ty(k) = Ny + 12 ; % Fixing the elevation
    Ny = Ty(k);
    Ty2(k) = Ny2 + 12 ; % No change to elevation 2
    Ny2 = Ty2(k);

end
Bx = [Tx, Tx2]; % Looping Bx, changes with every loop
By =  [Ty, Ty2];
```

```
figure (7)
scatter(Bx, By); % Working nicely
plot(Bx, By);

% ------------------------------------------
% PLOTTING POINTS FOR TRANSECT VERTICES ONLY
% ------------------------------------------
figure(8)
plot([FirstBatch.X], [FirstBatch.Y], 'k')
hold on
plot([Pre20thBatch.X], [Pre20thBatch.Y], 'k')
plot([Post20thBatch.X], [Post20thBatch.Y], 'k')
plot([LastBatch.X], [LastBatch.Y], 'k')
ax.XGrid = 'on'; % Toggle X Grid Lines
ax.YGrid = 'on'; % Y grid
axis manual % tight, fill, etc.
grid on
title('Tasman Glacier Terminus Positions February 2013 with Transect Points - March
2016')
xlabel ('X')
ylabel ('Y')
scatter(Bx, By, 'r', 'x'); % X green marker
hold off

% First batch Loop against 20 Transects 1 by 1
clear TransectX
clear TransectY
clear BatchX
clear VectorSize

% First Calculate the Vector Size of each Batch and store into a scalar
for i = 1:Size1(1);

        %if i == 9
        %continue % This will skip i = 9
        %end
BatchX = FirstBatch(i).X;
VectorSize = size(BatchX);
VectorScl(i) = VectorSize(2);
end
%
================================================================================
======================
% [1] Calculate the coordinates of X and Y for the first transect ONLY - NEED
% TO iterate each Tx and Ty coordinate with the vector size of each
% Terminus Positions in order to calculate intersections in the next step
for k = 1:20
TransectX = linspace(Tx(1), Tx2(1), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(1), Ty2(1), VectorScl(k));
TransectXFinal1{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal1{k} = TransectY;
end
% T2
for k = 1:20
TransectX = linspace(Tx(2), Tx2(2), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(2), Ty2(2), VectorScl(k));
TransectXFinal2{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal2{k} = TransectY;
end
% T3
for k = 1:20
TransectX = linspace(Tx(3), Tx2(3), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(3), Ty2(3), VectorScl(k));
TransectXFinal3{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal3{k} = TransectY;
```

```
end
% T4
for k = 1:20
TransectX = linspace(Tx(4), Tx2(4), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(4), Ty2(4), VectorScl(k));
TransectXFinal4{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal4{k} = TransectY;
end
% T5
for k = 1:20
TransectX = linspace(Tx(5), Tx2(5), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(5), Ty2(5), VectorScl(k));
TransectXFinal5{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal5{k} = TransectY;
end
% T6
for k = 1:20
TransectX = linspace(Tx(6), Tx2(6), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(6), Ty2(6), VectorScl(k));
TransectXFinal6{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal6{k} = TransectY;
end
% T7
for k = 1:20
TransectX = linspace(Tx(7), Tx2(7), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(7), Ty2(7), VectorScl(k));
TransectXFinal7{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal7{k} = TransectY;
end
% T8
for k = 1:20
TransectX = linspace(Tx(8), Tx2(8), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(8), Ty2(8), VectorScl(k));
TransectXFinal8{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal8{k} = TransectY;
end
% T9
for k = 1:20
TransectX = linspace(Tx(9), Tx2(9), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(9), Ty2(9), VectorScl(k));
TransectXFinal9{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal9{k} = TransectY;
end
% T10
for k = 1:20
TransectX = linspace(Tx(10), Tx2(10), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(10), Ty2(10), VectorScl(k));
TransectXFinal10{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal10{k} = TransectY;
end
% T11
for k = 1:20
TransectX = linspace(Tx(11), Tx2(11), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(11), Ty2(11), VectorScl(k));
TransectXFinal11{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal11{k} = TransectY;
end
% T12
for k = 1:20
```

```
TransectX = linspace(Tx(12), Tx2(12), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(12), Ty2(12), VectorScl(k));
TransectXFinal12{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal12{k} = TransectY;
end
% T13
for k = 1:20
TransectX = linspace(Tx(13), Tx2(13), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(13), Ty2(13), VectorScl(k));
TransectXFinal13{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal13{k} = TransectY;
end
% T14
for k = 1:20
TransectX = linspace(Tx(14), Tx2(14), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(14), Ty2(14), VectorScl(k));
TransectXFinal14{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal14{k} = TransectY;
end
% T15
for k = 1:20
TransectX = linspace(Tx(15), Tx2(15), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(15), Ty2(15), VectorScl(k));
TransectXFinal15{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal15{k} = TransectY;
end
% T16
for k = 1:20
TransectX = linspace(Tx(16), Tx2(16), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(16), Ty2(16), VectorScl(k));
TransectXFinal16{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal16{k} = TransectY;
end
% T17
for k = 1:20
TransectX = linspace(Tx(17), Tx2(17), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(17), Ty2(17), VectorScl(k));
TransectXFinal17{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal17{k} = TransectY;
end
% T18
for k = 1:20
TransectX = linspace(Tx(18), Tx2(18), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(18), Ty2(18), VectorScl(k));
TransectXFinal18{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal18{k} = TransectY;
end
% T19
for k = 1:20
TransectX = linspace(Tx(19), Tx2(19), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(19), Ty2(19), VectorScl(k));
TransectXFinal19{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal19{k} = TransectY;
end
% T20
for k = 1:20
TransectX = linspace(Tx(20), Tx2(20), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(20), Ty2(20), VectorScl(k));
TransectXFinal20{k} = TransectX; % WHAT IF I DONT STORE IT
```

```
TransectYFinal20{k} = TransectY;
end


%
================================================================================
======================
% [1] Calculate Intersections looping for k
% Due to unable to loop in Intersections.m
% PROBLEM IDENTIFIED IS THAT THE INTERSECTION SCRIPT REQUIRES THAT EACH
% TERMINUS POSITION LINE IS CHECKED FOR AN INTERSECTION WITH THE FIXED
% TRANSECT LINE. AS A RESULT WE NEED TO REPLOT FOR EACH TRANSECT THE NUMBER
% OF VECTORS... sigh. This is done and they are called TransectXFinal1-20
% and Y.
% T1
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal1(i)),
cell2mat(TransectYFinal1(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal1(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
AreaT1X1(i) = x1;
AreaT1Y1(i) = y1;


end
% T2 - ISSUE WITH Transect 2 and intersection. Could be a NON intercept at
% i = 2
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal2(i)),
cell2mat(TransectYFinal2(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal2(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T3
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal3(i)),
cell2mat(TransectYFinal3(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal3(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T4
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal4(i)),
cell2mat(TransectYFinal4(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal4(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T5
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal5(i)),
cell2mat(TransectYFinal5(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal5(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T6
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal6(i)),
cell2mat(TransectYFinal6(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal6(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T7
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal7(i)),
cell2mat(TransectYFinal7(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal7(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
```

```
% T8
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal8(i)),
cell2mat(TransectYFinal8(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal8(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T9
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal9(i)),
cell2mat(TransectYFinal9(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal9(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T10 - 1.4085 - Number 2
for i = 1:20;
    if i == 2
        continue
    end
    IntFinal10(2) = NaN;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal10(i)),
cell2mat(TransectYFinal10(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal10(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T11
for i = 1:20;
    %if i == 2
    %  continue
    %end
    %IntFinal11(2) = NaN; % Fill it with Nan rather than 0
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal11(i)),
cell2mat(TransectYFinal11(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal11(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T12
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal12(i)),
cell2mat(TransectYFinal12(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal12(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T13
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal13(i)),
cell2mat(TransectYFinal13(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal13(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T14 - Intersections fail at i = 2 (i.e. the 2nd terminus line did not
% intersect this)
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal14(i)),
cell2mat(TransectYFinal14(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal14(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world

end
% T15
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal15(i)),
cell2mat(TransectYFinal15(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal15(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T16
for i = 1:20;
```

```
    %if i == 2
     %   continue
    %end
     % Fill it with Nan rather than 0
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal16(i)),
cell2mat(TransectYFinal16(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works

IntFinal16(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
%IntFinal16(2) = NaN;
% T17 - 1.4085 - Number 3
for i = 1:20; % To fix problem. make sure you clear IntFinal17
    if i == 3;
       continue
    end
     % Fill it with Nan rather than 0
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal17(i)),
cell2mat(TransectYFinal17(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works

IntFinal17(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
IntFinal17(3) = NaN;
% T18
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal18(i)),
cell2mat(TransectYFinal18(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works

IntFinal18(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T19 - ISSUE TOO - Non intersection at 17.
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal19(i)),
cell2mat(TransectYFinal19(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
IntFinal19(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world

end
% T20
for i = 1:20;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal20(i)),
cell2mat(TransectYFinal20(i)), FirstBatch(i).X, FirstBatch(i).Y, 1); %It works
AreaT20X1(i) = x1;
AreaT20Y1(i) = y1;
IntFinal20(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%
% [EXTRA] PLOT TRANSECTS ON MAP
% Plotting test of each transect intersection for Batch One
% Assign all the Transects

for k = 1:20
TransectX = linspace(Tx(k), Tx2(k), 800); % Where 371 is the number of vector
points for FirstBatch(1).X*-
TransectY = linspace(Ty(k), Ty2(k), 800);
TransectXFinal{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal{k} = TransectY;
end

figure(9)
plot([FirstBatch.X], [FirstBatch.Y], 'k')
hold on
plot([Pre20thBatch.X], [Pre20thBatch.Y], 'k')
plot([Post20thBatch.X], [Post20thBatch.Y], 'k')
```

```
plot([LastBatch.X], [LastBatch.Y], 'k')
ax.XGrid = 'on'; % Toggle X Grid Lines
ax.YGrid = 'on'; % Y grid
axis manual % tight, fill, etc.
grid on
title('Tasman Glacier Terminus Positions with Transects 1 - 20')
xlabel ('X')
ylabel ('Y')
scatter(Bx, By, 'c', 'x'); % X green marker
% Plot individual Transects with colours
% Build a cmap first
Cscale = 1:20; % set number of colours here
cmap = buildcmap(Cscale);


plot(cell2mat(TransectXFinal(1)), cell2mat(TransectYFinal(1)), 'Color', cmap(1,:))
plot(cell2mat(TransectXFinal(2)), cell2mat(TransectYFinal(2)), 'Color', cmap(2,:))
plot(cell2mat(TransectXFinal(3)), cell2mat(TransectYFinal(3)), 'Color', cmap(3,:))
plot(cell2mat(TransectXFinal(4)), cell2mat(TransectYFinal(4)), 'Color', cmap(4,:))
plot(cell2mat(TransectXFinal(5)), cell2mat(TransectYFinal(5)), 'Color', cmap(5,:))
plot(cell2mat(TransectXFinal(6)), cell2mat(TransectYFinal(6)), 'Color', cmap(6,:))
plot(cell2mat(TransectXFinal(7)), cell2mat(TransectYFinal(7)), 'Color', cmap(7,:))
plot(cell2mat(TransectXFinal(8)), cell2mat(TransectYFinal(8)), 'Color', cmap(8,:))
plot(cell2mat(TransectXFinal(9)), cell2mat(TransectYFinal(9)), 'Color', cmap(9,:))
plot(cell2mat(TransectXFinal(10)), cell2mat(TransectYFinal(10)), 'Color',
cmap(10,:))
plot(cell2mat(TransectXFinal(11)), cell2mat(TransectYFinal(11)), 'Color',
cmap(11,:))
plot(cell2mat(TransectXFinal(12)), cell2mat(TransectYFinal(12)), 'Color',
cmap(12,:))
plot(cell2mat(TransectXFinal(13)), cell2mat(TransectYFinal(13)), 'Color',
cmap(13,:))
plot(cell2mat(TransectXFinal(14)), cell2mat(TransectYFinal(14)), 'Color',
cmap(14,:))
plot(cell2mat(TransectXFinal(15)), cell2mat(TransectYFinal(15)), 'Color',
cmap(15,:))
plot(cell2mat(TransectXFinal(16)), cell2mat(TransectYFinal(16)), 'Color',
cmap(16,:))
plot(cell2mat(TransectXFinal(17)), cell2mat(TransectYFinal(17)), 'Color',
cmap(17,:))
plot(cell2mat(TransectXFinal(18)), cell2mat(TransectYFinal(18)), 'Color',
cmap(18,:))
plot(cell2mat(TransectXFinal(19)), cell2mat(TransectYFinal(19)), 'Color',
cmap(19,:))
plot(cell2mat(TransectXFinal(20)), cell2mat(TransectYFinal(20)), 'Color',
cmap(20,:))

% Plot numbers over Transect
text(Lx, Ly, str); % Plot slope angle
for j = 1:20
    Mx = Tx2(j);
    My = Ty2(j);
    text(Mx, My, num2str(j)); % Plot Transect Number below the Transect
end
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
% [1] Plotting all Transect Intersections for Batch One
figure(10)
hold on

plot(Datenm1, IntFinalAv1, 'k') % Plots average
plot(Datenm1, IntFinal1,'Color', cmap(1,:))
plot(Datenm1, IntFinal2,'Color', cmap(2,:))
plot(Datenm1, IntFinal3,'Color', cmap(3,:))
plot(Datenm1, IntFinal4,'Color', cmap(4,:))
plot(Datenm1, IntFinal5,'Color', cmap(5,:))
plot(Datenm1, IntFinal6,'Color', cmap(6,:))
```

```
plot(Datenm1, IntFinal7,'Color', cmap(7,:))
plot(Datenm1, IntFinal8,'Color', cmap(8,:))
plot(Datenm1, IntFinal9,'Color', cmap(9,:))
plot(Datenm1, IntFinal10,'Color', cmap(10,:))
plot(Datenm1, IntFinal11,'Color', cmap(11,:))
plot(Datenm1, IntFinal12,'Color', cmap(12,:))
plot(Datenm1, IntFinal13,'Color', cmap(13,:))
plot(Datenm1, IntFinal14,'Color', cmap(14,:))
plot(Datenm1, IntFinal15,'Color', cmap(15,:)) % Almost identical to transect 1
plot(Datenm1, IntFinal16,'Color', cmap(16,:))
plot(Datenm1, IntFinal17,'Color', cmap(17,:))
plot(Datenm1, IntFinal18,'Color', cmap(18,:))
plot(Datenm1, IntFinal19,'Color', cmap(19,:))
plot(Datenm1, IntFinal20,'Color', cmap(20,:))
datetick ('x',  'dd-mmm-yyyy');
hold off
title ('Tasman Terminus Transect Positions - Batch One')
xlabel('Month and Year')
ylabel('Meters (m)')

% Plot Texts

text(Datenm1(1)-10, IntFinalAv1(1), 'Average');
text(Datenm1(20)-2, IntFinalAv1(20), 'Average');

text(Datenm1(1)-2, IntFinal1(1), num2str(1));
text(Datenm1(1)-2, IntFinal2(1), num2str(2));
text(Datenm1(1)-2, IntFinal3(1), num2str(3));
text(Datenm1(1)-2, IntFinal4(1), num2str(4));
text(Datenm1(1)-2, IntFinal5(1), num2str(5));
text(Datenm1(1)-2, IntFinal6(1), num2str(6));
text(Datenm1(1)-2, IntFinal7(1), num2str(7));
text(Datenm1(1)-2, IntFinal8(1), num2str(8));
text(Datenm1(1)-2, IntFinal9(1), num2str(9));
text(Datenm1(1)-2, IntFinal10(1), num2str(10));
text(Datenm1(1)-2, IntFinal11(1), num2str(11));
text(Datenm1(1)-2, IntFinal12(1), num2str(12));
text(Datenm1(1)-2, IntFinal13(1), num2str(13));
text(Datenm1(1)-2, IntFinal14(1), num2str(14));
text(Datenm1(1)-2, IntFinal15(1), num2str(15));
text(Datenm1(1)-2, IntFinal16(1), num2str(16));
text(Datenm1(1)-2, IntFinal17(1), num2str(17));
text(Datenm1(1)-2, IntFinal18(1), num2str(18));
text(Datenm1(1)-2, IntFinal19(1), num2str(19));
text(Datenm1(1)-2, IntFinal20(1), num2str(20));

%=========================================================================================
% [2] Pre 20th Batching
%
%=========================================================================================
clear TransectX
clear TransectY
clear BatchX
clear VectorSize

% First Calculate the Vector Size of each Batch and store into a scalar
for i = 1:Size2(1);
BatchX = Pre20thBatch(i).X;
VectorSize = size(BatchX);
VectorScl(i) = VectorSize(2);
end
%
%=========================================================================================
% [2] Calculate the coordinates of X and Y for the first transect ONLY - NEED
% TO iterate each Tx and Ty coordinate with the vector size of each
% Terminus Positions in order to calculate intersections in the next step
```

```
for k = 1:Size2(1)
TransectX = linspace(Tx(1), Tx2(1), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(1), Ty2(1), VectorScl(k));
TransectXFinal1{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal1{k} = TransectY;
end
% T2
for k = 1:Size2(1)
TransectX = linspace(Tx(2), Tx2(2), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(2), Ty2(2), VectorScl(k));
TransectXFinal2{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal2{k} = TransectY;
end
% T3
for k = 1:Size2(1)
TransectX = linspace(Tx(3), Tx2(3), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(3), Ty2(3), VectorScl(k));
TransectXFinal3{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal3{k} = TransectY;
end
% T4
for k = 1:Size2(1)
TransectX = linspace(Tx(4), Tx2(4), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(4), Ty2(4), VectorScl(k));
TransectXFinal4{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal4{k} = TransectY;
end
% T5
for k = 1:Size2(1)
TransectX = linspace(Tx(5), Tx2(5), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(5), Ty2(5), VectorScl(k));
TransectXFinal5{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal5{k} = TransectY;
end
% T6
for k = 1:Size2(1)
TransectX = linspace(Tx(6), Tx2(6), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(6), Ty2(6), VectorScl(k));
TransectXFinal6{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal6{k} = TransectY;
end
% T7
for k = 1:Size2(1)
TransectX = linspace(Tx(7), Tx2(7), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(7), Ty2(7), VectorScl(k));
TransectXFinal7{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal7{k} = TransectY;
end
% T8
for k = 1:Size2(1)
TransectX = linspace(Tx(8), Tx2(8), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(8), Ty2(8), VectorScl(k));
TransectXFinal8{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal8{k} = TransectY;
end
% T9
for k = 1:Size2(1)
TransectX = linspace(Tx(9), Tx2(9), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(9), Ty2(9), VectorScl(k));
```

```
TransectXFinal9{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal9{k} = TransectY;
end
% T10
for k = 1:Size2(1)
TransectX = linspace(Tx(10), Tx2(10), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(10), Ty2(10), VectorScl(k));
TransectXFinal10{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal10{k} = TransectY;
end
% T11
for k = 1:Size2(1)
TransectX = linspace(Tx(11), Tx2(11), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(11), Ty2(11), VectorScl(k));
TransectXFinal11{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal11{k} = TransectY;
end
% T12
for k = 1:Size2(1)
TransectX = linspace(Tx(12), Tx2(12), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(12), Ty2(12), VectorScl(k));
TransectXFinal12{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal12{k} = TransectY;
end
% T13
for k = 1:Size2(1)
TransectX = linspace(Tx(13), Tx2(13), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(13), Ty2(13), VectorScl(k));
TransectXFinal13{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal13{k} = TransectY;
end
% T14
for k = 1:Size2(1)
TransectX = linspace(Tx(14), Tx2(14), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(14), Ty2(14), VectorScl(k));
TransectXFinal14{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal14{k} = TransectY;
end
% T15
for k = 1:Size2(1)
TransectX = linspace(Tx(15), Tx2(15), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(15), Ty2(15), VectorScl(k));
TransectXFinal15{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal15{k} = TransectY;
end
% T16
for k = 1:Size2(1)
TransectX = linspace(Tx(16), Tx2(16), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(16), Ty2(16), VectorScl(k));
TransectXFinal16{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal16{k} = TransectY;
end
% T17
for k = 1:Size2(1)
TransectX = linspace(Tx(17), Tx2(17), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(17), Ty2(17), VectorScl(k));
TransectXFinal17{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal17{k} = TransectY;
end
% T18
```

```
for k = 1:Size2(1)
TransectX = linspace(Tx(18), Tx2(18), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(18), Ty2(18), VectorScl(k));
TransectXFinal18{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal18{k} = TransectY;
end
% T19
for k = 1:Size2(1)
TransectX = linspace(Tx(19), Tx2(19), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(19), Ty2(19), VectorScl(k));
TransectXFinal19{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal19{k} = TransectY;
end
% T20
for k = 1:Size2(1)
TransectX = linspace(Tx(20), Tx2(20), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(20), Ty2(20), VectorScl(k));
TransectXFinal20{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal20{k} = TransectY;
end


%
==========================================================================================
% Calculate Intersections looping for k
% Due to unable to loop in Intersections.m
% PROBLEM IDENTIFIED IS THAT THE INTERSECTION SCRIPT REQUIRES THAT EACH
% TERMINUS POSITION LINE IS CHECKED FOR AN INTERSECTION WITH THE FIXED
% TRANSECT LINE. AS A RESULT WE NEED TO REPLOT FOR EACH TRANSECT THE NUMBER
% OF VECTORS... sigh. This is done and they are called TransectXFinal1-20
% and Y.
% T1
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal1(i)),
cell2mat(TransectYFinal1(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
AreaT1X2(i) = x1;
AreaT1Y2(i) = y1;
IntFinalB1(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T2 - ISSUE WITH Transect 2 and intersection. Could be a NON intercept at
% i = 2
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal2(i)),
cell2mat(TransectYFinal2(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB2(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world

end

% T3
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal3(i)),
cell2mat(TransectYFinal3(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB3(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T4
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal4(i)),
cell2mat(TransectYFinal4(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB4(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
```

```
% T5
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal5(i)),
cell2mat(TransectYFinal5(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB5(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T6
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal6(i)),
cell2mat(TransectYFinal6(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB6(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T7
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal7(i)),
cell2mat(TransectYFinal7(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB7(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T8
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal8(i)),
cell2mat(TransectYFinal8(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB8(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T9
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal9(i)),
cell2mat(TransectYFinal9(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB9(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T10 - Issue with B10 at 21 (NEW)
for i = 1:Size2(1);
    % if i == 21
    %    continue
    %end
    %IntFinalB10(21) = NaN;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal10(i)),
cell2mat(TransectYFinal10(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB10(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T11 - Issue with B11 at 22
for i = 1:Size2(1);
    %if i == 22
    %    continue
    %end
    %IntFinalB11(22) = NaN;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal11(i)),
cell2mat(TransectYFinal11(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB11(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T12
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal12(i)),
cell2mat(TransectYFinal12(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB12(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T13
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal13(i)),
cell2mat(TransectYFinal13(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
```

```
IntFinalB13(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T14 - Intersections fail at i = 2 (i.e. the 2nd terminus line did not
% intersect this)
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal14(i)),
cell2mat(TransectYFinal14(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB14(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world

end
% T15
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal15(i)),
cell2mat(TransectYFinal15(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB15(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T16
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal16(i)),
cell2mat(TransectYFinal16(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB16(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T17
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal17(i)),
cell2mat(TransectYFinal17(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB17(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T18
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal18(i)),
cell2mat(TransectYFinal18(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB18(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T19 - ISSUE TOO - Non intersection at 17.
for i = 1:Size2(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal19(i)),
cell2mat(TransectYFinal19(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
IntFinalB19(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world

end
% T20 - Issue with 20 at 20,22 (NEW)
for i = 1:Size2(1);
    %    if i == 20
    %    continue
    %    end
    %if i == 22
    %    continue
    %end
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal20(i)),
cell2mat(TransectYFinal20(i)), Pre20thBatch(i).X, Pre20thBatch(i).Y, 1); %It works
AreaT20X2(i) = x1;
AreaT20Y2(i) = y1;
IntFinalB20(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
%IntFinalB20(20) = NaN;
%IntFinalB20(22) = NaN;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
% Average Ur
```

```
UrAv2 = 0.8528 * 365/121 % annual rate
UrAv2b = 0.8528 / 121 % daily rate
ErrorUrAv2 = ((IntFinalB18(1) - IntFinalB18(end)-15))* 365/121 % Standard error

% [2] Plotting all Transect Intersections for Batch Two
figure(11)
hold on
% Recreate Datenm1 for Size2(1)

plot(Datenm2, IntFinalAv2, 'k') % Plots average
plot(Datenm2, IntFinalB1,'Color', cmap(1,:))
plot(Datenm2, IntFinalB2,'Color', cmap(2,:))
plot(Datenm2, IntFinalB3,'Color', cmap(3,:))
plot(Datenm2, IntFinalB4,'Color', cmap(4,:))
plot(Datenm2, IntFinalB5,'Color', cmap(5,:))
plot(Datenm2, IntFinalB6,'Color', cmap(6,:))
plot(Datenm2, IntFinalB7,'Color', cmap(7,:))
plot(Datenm2, IntFinalB8,'Color', cmap(8,:))
plot(Datenm2, IntFinalB9,'Color', cmap(9,:))
plot(Datenm2, IntFinalB10,'Color', cmap(10,:))
plot(Datenm2, IntFinalB11,'Color', cmap(11,:))
plot(Datenm2, IntFinalB12,'Color', cmap(12,:))
plot(Datenm2, IntFinalB13,'Color', cmap(13,:))
plot(Datenm2, IntFinalB14,'Color', cmap(14,:))
plot(Datenm2, IntFinalB15,'Color', cmap(15,:)) % Almost identical to transect 1
plot(Datenm2, IntFinalB16,'Color', cmap(16,:))
plot(Datenm2, IntFinalB17,'Color', cmap(17,:))
plot(Datenm2, IntFinalB18,'Color', cmap(18,:))
plot(Datenm2, IntFinalB19,'Color', cmap(19,:))
plot(Datenm2, IntFinalB20,'Color', cmap(20,:))
datetick ('x',  'dd-mmm-yyyy');
hold off
title ('Tasman Terminus Transect Positions - Batch Two')
xlabel('Month and Year')
ylabel('Meters (m)')

% Plot Texts

text(Datenm2(1)-12, IntFinalAv2(1), 'Average')
text(Datenm2(end)+4, IntFinalAv2(end), 'Average')

text(Datenm2(1)-3, IntFinalB1(1), num2str(1));
text(Datenm2(1)-3, IntFinalB2(1), num2str(2));
text(Datenm2(1)-3, IntFinalB3(1), num2str(3));
text(Datenm2(1)-3, IntFinalB4(1), num2str(4));
text(Datenm2(1)-3, IntFinalB5(1), num2str(5));
text(Datenm2(1)-3, IntFinalB6(1), num2str(6));
text(Datenm2(1)-3, IntFinalB7(1), num2str(7));
text(Datenm2(1)-3, IntFinalB8(1), num2str(8));
text(Datenm2(1)-3, IntFinalB9(1), num2str(9));
text(Datenm2(1)-6, IntFinalB10(1), num2str(10));
text(Datenm2(1)-3, IntFinalB11(1), num2str(11));
text(Datenm2(1)-3, IntFinalB12(1), num2str(12));
text(Datenm2(1)-3, IntFinalB13(1), num2str(13));
text(Datenm2(1)-3, IntFinalB14(1), num2str(14));
text(Datenm2(1)-6, IntFinalB15(1), num2str(15));
text(Datenm2(1)-6, IntFinalB16(1), num2str(16));
text(Datenm2(1)-6, IntFinalB17(1), num2str(17));
text(Datenm2(1)-3, IntFinalB18(1), num2str(18));
text(Datenm2(1)-3, IntFinalB19(1), num2str(19));
text(Datenm2(1)-3, IntFinalB20(1), num2str(20));

%=========================================================================
===
% [3] Post 20th Batching
%
=========================================================================
=
clear TransectX
```

```
clear TransectY
clear BatchX
clear VectorSize

% First Calculate the Vector Size of each Batch and store into a scalar
for i = 1:Size3(1);
BatchX = Post20thBatch(i).X;
VectorSize = size(BatchX);
VectorScl(i) = VectorSize(2);
end
%
================================================================================
======================
% [3] Calculate the coordinates of X and Y for the first transect ONLY - NEED
% TO iterate each Tx and Ty coordinate with the vector size of each
% Terminus Positions in order to calculate intersections in the next step
for k = 1:Size3(1)
TransectX = linspace(Tx(1), Tx2(1), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(1), Ty2(1), VectorScl(k));
TransectXFinal1{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal1{k} = TransectY;
end
% T2
for k = 1:Size3(1)
TransectX = linspace(Tx(2), Tx2(2), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(2), Ty2(2), VectorScl(k));
TransectXFinal2{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal2{k} = TransectY;
end
% T3
for k = 1:Size3(1)
TransectX = linspace(Tx(3), Tx2(3), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(3), Ty2(3), VectorScl(k));
TransectXFinal3{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal3{k} = TransectY;
end
% T4
for k = 1:Size3(1)
TransectX = linspace(Tx(4), Tx2(4), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(4), Ty2(4), VectorScl(k));
TransectXFinal4{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal4{k} = TransectY;
end
% T5
for k = 1:Size3(1)
TransectX = linspace(Tx(5), Tx2(5), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(5), Ty2(5), VectorScl(k));
TransectXFinal5{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal5{k} = TransectY;
end
% T6
for k = 1:Size3(1)
TransectX = linspace(Tx(6), Tx2(6), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(6), Ty2(6), VectorScl(k));
TransectXFinal6{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal6{k} = TransectY;
end
% T7
for k = 1:Size3(1)
TransectX = linspace(Tx(7), Tx2(7), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(7), Ty2(7), VectorScl(k));
```

```
TransectXFinal7{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal7{k} = TransectY;
end
% T8
for k = 1:Size3(1)
TransectX = linspace(Tx(8), Tx2(8), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(8), Ty2(8), VectorScl(k));
TransectXFinal8{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal8{k} = TransectY;
end
% T9
for k = 1:Size3(1)
TransectX = linspace(Tx(9), Tx2(9), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(9), Ty2(9), VectorScl(k));
TransectXFinal9{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal9{k} = TransectY;
end
% T10
for k = 1:Size3(1)
TransectX = linspace(Tx(10), Tx2(10), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(10), Ty2(10), VectorScl(k));
TransectXFinal10{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal10{k} = TransectY;
end
% T11
for k = 1:Size3(1)
TransectX = linspace(Tx(11), Tx2(11), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(11), Ty2(11), VectorScl(k));
TransectXFinal11{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal11{k} = TransectY;
end
% T12
for k = 1:Size3(1)
TransectX = linspace(Tx(12), Tx2(12), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(12), Ty2(12), VectorScl(k));
TransectXFinal12{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal12{k} = TransectY;
end
% T13
for k = 1:Size3(1)
TransectX = linspace(Tx(13), Tx2(13), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(13), Ty2(13), VectorScl(k));
TransectXFinal13{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal13{k} = TransectY;
end
% T14
for k = 1:Size3(1)
TransectX = linspace(Tx(14), Tx2(14), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(14), Ty2(14), VectorScl(k));
TransectXFinal14{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal14{k} = TransectY;
end
% T15
for k = 1:Size3(1)
TransectX = linspace(Tx(15), Tx2(15), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(15), Ty2(15), VectorScl(k));
TransectXFinal15{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal15{k} = TransectY;
end
% T16
```

```
for k = 1:Size3(1)
TransectX = linspace(Tx(16), Tx2(16), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(16), Ty2(16), VectorScl(k));
TransectXFinal16{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal16{k} = TransectY;
end
% T17
for k = 1:Size3(1)
TransectX = linspace(Tx(17), Tx2(17), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(17), Ty2(17), VectorScl(k));
TransectXFinal17{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal17{k} = TransectY;
end
% T18
for k = 1:Size3(1)
TransectX = linspace(Tx(18), Tx2(18), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(18), Ty2(18), VectorScl(k));
TransectXFinal18{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal18{k} = TransectY;
end
% T19
for k = 1:Size3(1)
TransectX = linspace(Tx(19), Tx2(19), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(19), Ty2(19), VectorScl(k));
TransectXFinal19{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal19{k} = TransectY;
end
% T20
for k = 1:Size3(1)
TransectX = linspace(Tx(20), Tx2(20), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(20), Ty2(20), VectorScl(k));
TransectXFinal20{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal20{k} = TransectY;
end


%
================================================================================
======================
% [3] Calculate Intersections looping for k
% T1
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal1(i)),
cell2mat(TransectYFinal1(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It works
AreaT1X3(i) = x1;
AreaT1Y3(i) = y1;
IntFinalC1(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T2 - ISSUE WITH Transect 2 and intersection. Could be a NON intercept at
% i = 2
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal2(i)),
cell2mat(TransectYFinal2(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It works
IntFinalC2(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T3 - #74 non  for 1.39
for i = 1:Size3(1); % Number 74 not intersecting
    %if i == 74
     %   continue
    %end
    %IntFinalC3(74) = NaN;
```

```
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal3(i)),
cell2mat(TransectYFinal3(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It works
IntFinalC3(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T4 -#17 non intercept - ELABORATE ON THIS SPECIFICALLY IN THESIS -
% testing for non intercepts
for i = 1:Size3(1);
    %if i == 17
    %    continue
    %end
    %IntFinalC4(17) = NaN;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal4(i)),
cell2mat(TransectYFinal4(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It works
IntFinalC4(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T5
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal5(i)),
cell2mat(TransectYFinal5(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It works
IntFinalC5(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T6
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal6(i)),
cell2mat(TransectYFinal6(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It works
IntFinalC6(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T7
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal7(i)),
cell2mat(TransectYFinal7(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It works
IntFinalC7(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T8
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal8(i)),
cell2mat(TransectYFinal8(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It works
IntFinalC8(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T9
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal9(i)),
cell2mat(TransectYFinal9(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It works
IntFinalC9(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T10
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal10(i)),
cell2mat(TransectYFinal10(i)), Post20thBatch(i).X,Post20thBatch(i).Y, 1); %It works
IntFinalC10(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T11
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal11(i)),
cell2mat(TransectYFinal11(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It
works
IntFinalC11(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T12
```

```
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal12(i)),
cell2mat(TransectYFinal12(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It
works
IntFinalC12(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T13
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal13(i)),
cell2mat(TransectYFinal13(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It
works
IntFinalC13(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T14 - Intersections fail at i = 2 (i.e. the 2nd terminus line did not
% intersect this)
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal14(i)),
cell2mat(TransectYFinal14(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It
works
IntFinalC14(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world

end
% T15
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal15(i)),
cell2mat(TransectYFinal15(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It
works
IntFinalC15(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T16
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal16(i)),
cell2mat(TransectYFinal16(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It
works
IntFinalC16(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T17 - (NEW) Issue at 44, 1.4085 - Number 43
for i = 1:Size3(1);
    if i == 43
        continue
    end
    IntFinalC17(43) = NaN;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal17(i)),
cell2mat(TransectYFinal17(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It
works
IntFinalC17(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T18
for i = 1:Size3(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal18(i)),
cell2mat(TransectYFinal18(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It
works
IntFinalC18(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T19 - (NEW) Issue at 30, 31, 32,33, (NEWER) 1.4085 - Number 21-27
for i = 1:Size3(1);
    if i == 21
        continue
    end
    if i == 22
        continue
```

```
    end
    if i == 23
        continue
    end
    if i == 24
        continue
    end
     if i == 25
        continue
     end
        if i == 26
        continue
        end
    if i == 27
        continue
    end
    IntFinalC19(21) = NaN;
    IntFinalC19(22) = NaN;
    IntFinalC19(23) = NaN;
    IntFinalC19(24) = NaN;
    IntFinalC19(25) = NaN;
    IntFinalC19(26) = NaN;
    IntFinalC19(27) = NaN;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal19(i)),
cell2mat(TransectYFinal19(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It
works
IntFinalC19(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T20 - (NEW) at 17
for i = 1:Size3(1);
      %if i == 17
      %  continue
      %end
    %IntFinalC20(17) = NaN;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal20(i)),
cell2mat(TransectYFinal20(i)), Post20thBatch(i).X, Post20thBatch(i).Y, 1); %It
works
AreaT20X3(i) = x1;
AreaT20Y3(i) = y1;
IntFinalC20(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%% Average Ur
UrAv3 = (IntFinalAv3 (1) - IntFinalAv3(end) - 12) * 365/639;
UrDay3 = (IntFinalAv3 (1) - IntFinalAv3(end))/639

% MAX Ur for Standard error
MaxUr3 = (IntFinalC17(1) - IntFinalC17(end)-12)* 365/639
MinUr3 = (IntFinalC11(1) - IntFinalC11(end)-12)* 365/639
ErrUr3 = (IntFinalC17(1) - IntFinalC17(end)-12 -StdErr3)* 365/639 - UrAv3

ErrorUr3 = (12.5+5.23)*365/639 %10.01
%% Error for Ur Av
Error3 = 16.01;

UrAv3b = ((IntFinalC2(1) - IntFinalC2(end))+Error3) * 365/639 % Highest deviation
therefore highest Ur for max

% [3] Plotting all Transect Intersections for Batch Two - Use datenm2
figure(13)
hold on


plot(Datenm3, IntFinalC1,'Color', cmap(1,:))
plot(Datenm3, IntFinalC2,'Color', cmap(2,:))
```

```
plot(Datenm3, IntFinalC3,'Color', cmap(3,:))
plot(Datenm3, IntFinalC4,'Color', cmap(4,:))
plot(Datenm3, IntFinalC5,'Color', cmap(5,:))
plot(Datenm3, IntFinalC6,'Color', cmap(6,:))
plot(Datenm3, IntFinalC7,'Color', cmap(7,:))
plot(Datenm3, IntFinalC8,'Color', cmap(8,:))
plot(Datenm3, IntFinalC9,'Color', cmap(9,:))
plot(Datenm3, IntFinalC10,'Color', cmap(10,:))
plot(Datenm3, IntFinalC11,'Color', cmap(11,:))
plot(Datenm3, IntFinalC12,'Color', cmap(12,:))
plot(Datenm3, IntFinalC13,'Color', cmap(13,:))
plot(Datenm3, IntFinalC14,'Color', cmap(14,:))
plot(Datenm3, IntFinalC15,'Color', cmap(15,:)) % Almost identical to transect 1
plot(Datenm3, IntFinalC16,'Color', cmap(16,:))
plot(Datenm3, IntFinalC17,'Color', cmap(17,:))
plot(Datenm3, IntFinalC18,'Color', cmap(18,:))
plot(Datenm3, IntFinalC19,'Color', cmap(19,:))
plot(Datenm3, IntFinalC20,'Color', cmap(20,:))
plot(Datenm3, IntFinalAv3, 'k')
datetick ('x',  'dd-mmm-yyyy');
xlim([Datenm3(1)-120 Datenm3(end)+100])
hold off
title ('Tasman Terminus Transect Positions - Batch Three')
xlabel('Month and Year')
ylabel('Meters (m)')
% Plot Texts

text(Datenm3(1)-100, IntFinalAv3(1), 'Average');
text(Datenm3(end), IntFinalAv3(end), 'Average');

text(Datenm3(1)-10, IntFinalC1(1), num2str(1));
text(Datenm3(1)-10, IntFinalC2(1), num2str(2));
text(Datenm3(1)-10, IntFinalC3(1), num2str(3));
text(Datenm3(1)-20, IntFinalC4(1), num2str(4));
text(Datenm3(1)-20, IntFinalC5(1), num2str(5));
text(Datenm3(1)-20, IntFinalC6(1), num2str(6));
text(Datenm3(1)-30, IntFinalC7(1), num2str(7));
text(Datenm3(1)-30, IntFinalC8(1), num2str(8));
text(Datenm3(1)-30, IntFinalC9(1), num2str(9));
text(Datenm3(1)-30, IntFinalC10(1), num2str(10));
text(Datenm3(1)-30, IntFinalC11(1), num2str(11));
text(Datenm3(1)-40, IntFinalC12(1), num2str(12));
text(Datenm3(1)-40, IntFinalC13(1), num2str(13));
text(Datenm3(1)-40, IntFinalC14(1), num2str(14));
text(Datenm3(1)-50, IntFinalC15(1), num2str(15));
text(Datenm3(1)-50, IntFinalC16(1), num2str(16));
text(Datenm3(1)-50, IntFinalC17(1), num2str(17));
text(Datenm3(1)-20, IntFinalC18(1), num2str(18));
text(Datenm3(1)-20, IntFinalC19(1), num2str(19));
text(Datenm3(1)-20, IntFinalC20(1), num2str(20));


%=========================================================================================
===
% [4] Last Batch Batching
%
=========================================================================================
=
clear TransectX
clear TransectY
clear BatchX
clear VectorSize

% First Calculate the Vector Size of each Batch and store into a scalar
for i = 1:Size4(1);

        %if i == 9
        %continue % This will skip i = 9
        %end
```

```
BatchX = Post20thBatch(i).X;
VectorSize = size(BatchX);
VectorScl(i) = VectorSize(2);
end
%
================================================================================================
% [4] Calculate the coordinates of X and Y for the first transect ONLY - NEED
% TO iterate each Tx and Ty coordinate with the vector size of each
% Terminus Positions in order to calculate intersections in the next step
for k = 1:Size4(1)
TransectX = linspace(Tx(1), Tx2(1), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(1), Ty2(1), VectorScl(k));
TransectXFinal1{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal1{k} = TransectY;
end
% T2
for k = 1:Size4(1)
TransectX = linspace(Tx(2), Tx2(2), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(2), Ty2(2), VectorScl(k));
TransectXFinal2{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal2{k} = TransectY;
end
% T3
for k = 1:Size4(1)
TransectX = linspace(Tx(3), Tx2(3), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(3), Ty2(3), VectorScl(k));
TransectXFinal3{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal3{k} = TransectY;
end
% T4
for k = 1:Size4(1)
TransectX = linspace(Tx(4), Tx2(4), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(4), Ty2(4), VectorScl(k));
TransectXFinal4{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal4{k} = TransectY;
end
% T5
for k = 1:Size4(1)
TransectX = linspace(Tx(5), Tx2(5), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(5), Ty2(5), VectorScl(k));
TransectXFinal5{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal5{k} = TransectY;
end
% T6
for k = 1:Size4(1)
TransectX = linspace(Tx(6), Tx2(6), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(6), Ty2(6), VectorScl(k));
TransectXFinal6{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal6{k} = TransectY;
end
% T7
for k = 1:Size4(1)
TransectX = linspace(Tx(7), Tx2(7), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(7), Ty2(7), VectorScl(k));
TransectXFinal7{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal7{k} = TransectY;
end
% T8
for k = 1:Size4(1)
```

```
TransectX = linspace(Tx(8), Tx2(8), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(8), Ty2(8), VectorScl(k));
TransectXFinal8{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal8{k} = TransectY;
end
% T9
for k = 1:Size4(1)
TransectX = linspace(Tx(9), Tx2(9), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(9), Ty2(9), VectorScl(k));
TransectXFinal9{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal9{k} = TransectY;
end
% T10
for k = 1:Size4(1)
TransectX = linspace(Tx(10), Tx2(10), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(10), Ty2(10), VectorScl(k));
TransectXFinal10{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal10{k} = TransectY;
end
% T11
for k = 1:Size4(1)
TransectX = linspace(Tx(11), Tx2(11), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(11), Ty2(11), VectorScl(k));
TransectXFinal11{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal11{k} = TransectY;
end
% T12
for k = 1:Size4(1)
TransectX = linspace(Tx(12), Tx2(12), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(12), Ty2(12), VectorScl(k));
TransectXFinal12{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal12{k} = TransectY;
end
% T13
for k = 1:Size4(1)
TransectX = linspace(Tx(13), Tx2(13), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(13), Ty2(13), VectorScl(k));
TransectXFinal13{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal13{k} = TransectY;
end
% T14
for k = 1:Size4(1)
TransectX = linspace(Tx(14), Tx2(14), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(14), Ty2(14), VectorScl(k));
TransectXFinal14{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal14{k} = TransectY;
end
% T15
for k = 1:Size4(1)
TransectX = linspace(Tx(15), Tx2(15), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(15), Ty2(15), VectorScl(k));
TransectXFinal15{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal15{k} = TransectY;
end
% T16
for k = 1:Size4(1)
TransectX = linspace(Tx(16), Tx2(16), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(16), Ty2(16), VectorScl(k));
TransectXFinal16{k} = TransectX; % WHAT IF I DONT STORE IT
```

```
TransectYFinal16{k} = TransectY;
end
% T17
for k = 1:Size4(1)
TransectX = linspace(Tx(17), Tx2(17), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(17), Ty2(17), VectorScl(k));
TransectXFinal17{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal17{k} = TransectY;
end
% T18
for k = 1:Size4(1)
TransectX = linspace(Tx(18), Tx2(18), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(18), Ty2(18), VectorScl(k));
TransectXFinal18{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal18{k} = TransectY;
end
% T19
for k = 1:Size4(1)
TransectX = linspace(Tx(19), Tx2(19), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(19), Ty2(19), VectorScl(k));
TransectXFinal19{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal19{k} = TransectY;
end
% T20
for k = 1:Size4(1)
TransectX = linspace(Tx(20), Tx2(20), VectorScl(k)); % Where 371 is the number of
vector points for FirstBatch(1).X*-
TransectY = linspace(Ty(20), Ty2(20), VectorScl(k));
TransectXFinal20{k} = TransectX; % WHAT IF I DONT STORE IT
TransectYFinal20{k} = TransectY;
end


%
================================================================================
======================
% [4] Calculate Intersections looping for k - No issues for 1.4085
% T1
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal1(i)),
cell2mat(TransectYFinal1(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
AreaT1X4(i) = x1;
AreaT1Y4(i) = y1;
IntFinalD1(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T2 - ISSUE WITH Transect 2 and intersection. Could be a NON intercept at
% i = 2
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal2(i)),
cell2mat(TransectYFinal2(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD2(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T3 - #74 non intersect
for i = 1:Size4(1); % Number 74 not intersecting
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal3(i)),
cell2mat(TransectYFinal3(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD3(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T4 -#17 non intercept - ELABORATE ON THIS SPECIFICALLY IN THESIS -
% testing for non intercepts
for i = 1:Size4(1);
```

```
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal4(i)),
cell2mat(TransectYFinal4(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD4(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T5
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal5(i)),
cell2mat(TransectYFinal5(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD5(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T6
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal6(i)),
cell2mat(TransectYFinal6(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD6(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T7
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal7(i)),
cell2mat(TransectYFinal7(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD7(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T8
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal8(i)),
cell2mat(TransectYFinal8(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD8(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T9
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal9(i)),
cell2mat(TransectYFinal9(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD9(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T10
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal10(i)),
cell2mat(TransectYFinal10(i)), LastBatch(i).X,LastBatch(i).Y, 1); %It works
IntFinalD10(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T11
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal11(i)),
cell2mat(TransectYFinal11(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD11(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T12
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal12(i)),
cell2mat(TransectYFinal12(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD12(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T13
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal13(i)),
cell2mat(TransectYFinal13(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD13(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
```

```
% T14 - Intersections fail at i = 2 (i.e. the 2nd terminus line did not
% intersect this)
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal14(i)),
cell2mat(TransectYFinal14(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD14(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world

end
% T15
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal15(i)),
cell2mat(TransectYFinal15(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD15(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T16
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal16(i)),
cell2mat(TransectYFinal16(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD16(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T17
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal17(i)),
cell2mat(TransectYFinal17(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD17(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T18 - (NEW) Issue at 14
for i = 1:Size4(1);
    %if i == 14
    %    continue
    %end
    %IntFinalD18(14) = NaN;
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal18(i)),
cell2mat(TransectYFinal18(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD18(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end
% T19 - ISSUE TOO - Non intersection at 17.
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal19(i)),
cell2mat(TransectYFinal19(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
IntFinalD19(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world

end
% T20
for i = 1:Size4(1);
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal20(i)),
cell2mat(TransectYFinal20(i)), LastBatch(i).X, LastBatch(i).Y, 1); %It works
AreaT20X4(i) = x1;
AreaT20Y4(i) = y1;
IntFinalD20(i) = iout1 * (Dist/VectorScl(i)); % Output of where it is on the line .
DIST - is the distance of the transect in real world
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%%% Calculate Ur %%%%%
UrAv4 = (IntFinalAv4(1) - IntFinalAv4(end)- 16)* 365/208
DayAv4 = (IntFinalAv4(1) - IntFinalAv4(end)) / 208
ErrorUr4 = (16 + StdErr4)*365/208 % 54.26

MaxUr4 = (IntFinalD5(1) - IntFinalD5(end)- 16)* 365/208 % 459.34
MinUr4 = (IntFinalD20(1)- IntFinalD20(end)- 16)* 365/208 % 115.92
```

```
%%% Errors
MaxUr4 = (IntFinalD4(1) - IntFinalD4(end)+15)* 365/208



%%%%%%
% [4] Plotting all Transect Intersections for Batch Four - Use datenm2

% REMOVE First Two Entries of every Intersection
IntFinalD1 = IntFinalD1(3:end);
IntFinalD2 = IntFinalD2(3:end);
IntFinalD3 = IntFinalD3(3:end);
IntFinalD4 = IntFinalD4(3:end);
IntFinalD5 = IntFinalD5(3:end);
IntFinalD6 = IntFinalD6(3:end);
IntFinalD7 = IntFinalD7(3:end);
IntFinalD8 = IntFinalD8(3:end);
IntFinalD9 = IntFinalD9(3:end);
IntFinalD10 = IntFinalD10(3:end);
IntFinalD11 = IntFinalD11(3:end);
IntFinalD12 = IntFinalD12(3:end);
IntFinalD13 = IntFinalD13(3:end);
IntFinalD14 = IntFinalD14(3:end);
IntFinalD15 = IntFinalD15(3:end);
IntFinalD16 = IntFinalD16(3:end);
IntFinalD17 = IntFinalD17(3:end);
IntFinalD18 = IntFinalD18(3:end);
IntFinalD19 = IntFinalD19(3:end);
IntFinalD20 = IntFinalD20(3:end);

%%%%% PLOT %%%%
figure(14)
hold on
plot(Datenm4, IntFinalD2,'Color', cmap(2,:))
plot(Datenm4, IntFinalD3,'Color', cmap(3,:))
plot(Datenm4, IntFinalD4,'Color', cmap(4,:))
plot(Datenm4, IntFinalD5,'Color', cmap(5,:))
plot(Datenm4, IntFinalD6,'Color', cmap(6,:))
plot(Datenm4, IntFinalD7,'Color', cmap(7,:))
plot(Datenm4, IntFinalD8,'Color', cmap(8,:))
plot(Datenm4, IntFinalD9,'Color', cmap(9,:))
plot(Datenm4, IntFinalD10,'Color', cmap(10,:))
plot(Datenm4, IntFinalD11,'Color', cmap(11,:))
plot(Datenm4, IntFinalD12,'Color', cmap(12,:))
plot(Datenm4, IntFinalD13,'Color', cmap(13,:))
plot(Datenm4, IntFinalD14,'Color', cmap(14,:))
plot(Datenm4, IntFinalD15,'Color', cmap(15,:)) % Almost identical to transect 1
plot(Datenm4, IntFinalD16,'Color', cmap(16,:))
plot(Datenm4, IntFinalD17,'Color', cmap(17,:))
plot(Datenm4, IntFinalD18,'Color', cmap(18,:))
plot(Datenm4, IntFinalD19,'Color', cmap(19,:))
plot(Datenm4, IntFinalD20,'Color', cmap(20,:))
plot(Datenm4, IntFinalAv4, 'k');

datetick ('x',  'dd-mmm-yyyy');
hold off
title ('Tasman Terminus Transect Positions - Batch Four')
xlabel('Month and Year')
ylabel('Meters (m)')
xlim([Datenm4(1)-10 Datenm4(end)+5])
% Plot Texts

text(Datenm4(1)-11, IntFinalAv4(1), 'Average');
text(Datenm4(end)+2, IntFinalAv4(end), 'Average');

text(Datenm4(1)-1, IntFinalD1(1), num2str(1));
text(Datenm4(1)-1, IntFinalD2(1), num2str(2));
```

```
text(Datenm4(1)-1, IntFinalD3(1), num2str(3));
text(Datenm4(1)-3, IntFinalD4(1), num2str(4));
text(Datenm4(1)-3, IntFinalD5(1), num2str(5));
text(Datenm4(1)-3, IntFinalD6(1), num2str(6));
text(Datenm4(1)-1, IntFinalD7(1), num2str(7));
text(Datenm4(1)-1, IntFinalD8(1), num2str(8));
text(Datenm4(1)-1, IntFinalD9(1), num2str(9));
text(Datenm4(1)-1, IntFinalD10(1), num2str(10));
text(Datenm4(1)-5, IntFinalD11(1), num2str(11));
text(Datenm4(1)-5, IntFinalD12(1), num2str(12));
text(Datenm4(1)-7, IntFinalD13(1), num2str(13));
text(Datenm4(1)-7, IntFinalD14(1), num2str(14));
text(Datenm4(1)-7, IntFinalD15(1), num2str(15));
text(Datenm4(1)-3, IntFinalD16(1), num2str(16));
text(Datenm4(1)-3, IntFinalD17(1), num2str(17));
text(Datenm4(1)-3, IntFinalD18(1), num2str(18));
text(Datenm4(1)-3, IntFinalD19(1), num2str(19));
text(Datenm4(1)-3, IntFinalD20(1), num2str(20));


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
% [5] Plotting ALL Transect Results
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
figure(15)
hold on

IntAll1 = [IntFinal1, IntFinalB1, IntFinalC1, IntFinalD1]; % Combined Intersections
IntAll2 = [IntFinal2, IntFinalB2, IntFinalC2, IntFinalD2]; % Combined Intersections
IntAll3 = [IntFinal3, IntFinalB3, IntFinalC3, IntFinalD3]; % Combined Intersections
IntAll4 = [IntFinal4, IntFinalB4, IntFinalC4, IntFinalD4]; % Combined Intersections
IntAll5 = [IntFinal5, IntFinalB5, IntFinalC5, IntFinalD5]; % Combined Intersections
IntAll6 = [IntFinal6, IntFinalB6, IntFinalC6, IntFinalD6]; % Combined Intersections
IntAll7 = [IntFinal7, IntFinalB7, IntFinalC7, IntFinalD7]; % Combined Intersections
IntAll8 = [IntFinal8, IntFinalB8, IntFinalC8, IntFinalD8]; % Combined Intersections
IntAll9 = [IntFinal9, IntFinalB9, IntFinalC9, IntFinalD9]; % Combined Intersections
IntAll10 = [IntFinal10, IntFinalB10, IntFinalC10, IntFinalD10]; % Combined
Intersections
IntAll11 = [IntFinal11, IntFinalB11, IntFinalC11, IntFinalD11]; % Combined
Intersections
IntAll12 = [IntFinal12, IntFinalB12, IntFinalC12, IntFinalD12]; % Combined
Intersections
IntAll13 = [IntFinal13, IntFinalB13, IntFinalC13, IntFinalD13]; % Combined
Intersections
IntAll14 = [IntFinal14, IntFinalB14, IntFinalC14, IntFinalD14]; % Combined
Intersections
IntAll15 = [IntFinal15, IntFinalB15, IntFinalC15, IntFinalD15]; % Combined
Intersections
IntAll16 = [IntFinal16, IntFinalB16, IntFinalC16, IntFinalD16]; % Combined
Intersections
IntAll17 = [IntFinal17, IntFinalB17, IntFinalC17, IntFinalD17]; % Combined
Intersections
IntAll18 = [IntFinal18, IntFinalB18, IntFinalC18, IntFinalD18]; % Combined
Intersections
IntAll19 = [IntFinal19, IntFinalB19, IntFinalC19, IntFinalD19]; % Combined
Intersections
IntAll20 = [IntFinal20, IntFinalB20, IntFinalC20, IntFinalD20]; % Combined
Intersections

plot(DateAll1, IntAll1,'Color', cmap(1,:))
plot(DateAll1, IntAll2,'Color', cmap(2,:))
plot(DateAll1, IntAll3,'Color', cmap(3,:))
plot(DateAll1, IntAll4,'Color', cmap(4,:))
plot(DateAll1, IntAll5,'Color', cmap(5,:))
plot(DateAll1, IntAll6,'Color', cmap(6,:))
plot(DateAll1, IntAll7,'Color', cmap(7,:))
plot(DateAll1, IntAll8,'Color', cmap(8,:))
```

```
plot(DateAll1, IntAll9,'Color', cmap(9,:))
plot(DateAll1, IntAll10,'Color', cmap(10,:))
plot(DateAll1, IntAll11,'Color', cmap(11,:))
plot(DateAll1, IntAll12,'Color', cmap(12,:))
plot(DateAll1, IntAll13,'Color', cmap(13,:))
plot(DateAll1, IntAll14,'Color', cmap(14,:))
plot(DateAll1, IntAll15,'Color', cmap(15,:)) % Almost identical to transect 1
plot(DateAll1, IntAll16,'Color', cmap(16,:))
plot(DateAll1, IntAll17,'Color', cmap(17,:))
plot(DateAll1, IntAll18,'Color', cmap(18,:))
plot(DateAll1, IntAll19,'Color', cmap(19,:))
plot(DateAll1, IntAll20,'Color', cmap(20,:))
datetick ('x',  'dd-mmm-yyyy');
hold off
title ('Tasman Terminus Transect Positions - All batches')
xlabel('Month and Year')
ylabel('Meters (m)')

% Plot Texts

text(Datenm4(1), IntFinalD1(1), num2str(1));
text(Datenm4(1), IntFinalD2(1), num2str(2));
text(Datenm4(1), IntFinalD3(1), num2str(3));
text(Datenm4(1), IntFinalD4(1), num2str(4));
text(Datenm4(1), IntFinalD5(1), num2str(5));
text(Datenm4(1), IntFinalD6(1), num2str(6));
text(Datenm4(1), IntFinalD7(1), num2str(7));
text(Datenm4(1), IntFinalD8(1), num2str(8));
text(Datenm4(1), IntFinalD9(1), num2str(9));
text(Datenm4(1), IntFinalD10(1), num2str(10));
text(Datenm4(1), IntFinalD11(1), num2str(11));
text(Datenm4(1), IntFinalD12(1), num2str(12));
text(Datenm4(1), IntFinalD13(1), num2str(13));
text(Datenm4(1), IntFinalD14(1), num2str(14));
text(Datenm4(1), IntFinalD15(1), num2str(15));
text(Datenm4(1), IntFinalD16(1), num2str(16));
text(Datenm4(1), IntFinalD17(1), num2str(17));
text(Datenm4(1), IntFinalD18(1), num2str(18));
text(Datenm4(1), IntFinalD19(1), num2str(19));
text(Datenm4(1), IntFinalD20(1), num2str(20));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (6) Average all intersections and plot as a single timeseries
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

IntAllAverage =
(IntAll1+IntAll2+IntAll3+IntAll4+IntAll5+IntAll6+IntAll7+IntAll8+IntAll9+IntAll10+I
ntAll11+IntAll12+IntAll13+IntAll14+IntAll15+IntAll16+IntAll17+IntAll18+IntAll19+Int
All20)/20;
% For intersections with NaN - recalculate
% Second Entry - IntAll10
IntAllAverage(2) =
(IntAll1(2)+IntAll2(2)+IntAll3(2)+IntAll4(2)+IntAll5(2)+IntAll6(2)+IntAll7(2)+IntAl
l8(2)+IntAll9(2)+IntAll11(2)+IntAll12(2)+IntAll13(2)+IntAll14(2)+IntAll15(2)+IntAll
16(2)+IntAll17(2)+IntAll18(2)+IntAll19(2)+IntAll20(2))/19;
disp(IntAllAverage(2))
% Third Entry - IntAll17 has a NaN - identified and omit from average
IntAllAverage(3) =
(IntAll1(3)+IntAll2(3)+IntAll3(3)+IntAll4(3)+IntAll5(3)+IntAll6(3)+IntAll7(3)+IntAl
l8(3)+IntAll9(3)+IntAll10(3)+IntAll11(3)+IntAll12(3)+IntAll13(3)+IntAll14(3)+IntAll
15(3)+IntAll16(3)+IntAll18(3)+IntAll19(3)+IntAll20(3))/19;
disp(IntAllAverage(3))
% 66-72 Entries - All NaN in IntAll19
for i = 66:72
IntAllAverage(i) =
(IntAll1(i)+IntAll2(i)+IntAll3(i)+IntAll4(i)+IntAll5(i)+IntAll6(i)+IntAll7(i)+IntAl
```

```
l8(i)+IntAll9(i)+IntAll10(i)+IntAll11(i)+IntAll12(i)+IntAll13(i)+IntAll14(i)+IntAll
15(i)+IntAll16(i)+IntAll17(i)+IntAll18(i)+IntAll20(i))/19;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% AVERAGE RATES OF RETREAT FROM EACH BATCH %%%%%%%%%%%%%%%%
%%% BATCH ONE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

IntAll17(3) = 0; % Removing NaNs
IntAll10(2) = 0; % Ditto
IntFinalAv1 =
(IntAll1(1:20)+IntAll2(1:20)+IntAll3(1:20)+IntAll14(1:20)+IntAll5(1:20)+IntAll6(1:2
0)+IntAll7(1:20)+IntAll8(1:20)+IntAll9(1:20)+IntAll10(1:20)+IntAll11(1:20)+IntAll12
(1:20)+IntAll13(1:20)+IntAll14(1:20)+IntAll15(1:20)+IntAll16(1:20)+IntAll17(1:20)+I
ntAll18(1:20)+IntAll19(1:20)+IntAll20(1:20))/20
IntFinalAv1(2) =
(IntAll1(2)+IntAll2(2)+IntAll3(2)+IntAll14(2)+IntAll5(2)+IntAll6(2)+IntAll7(2)+IntA
ll8(2)+IntAll9(2)+IntAll10(2)+IntAll11(2)+IntAll12(2)+IntAll13(2)+IntAll14(2)+IntAl
l15(2)+IntAll16(2)+IntAll17(2)+IntAll18(2)+IntAll19(2)+IntAll20(2))/19 % Manual
average due to NaN in IntAll10
IntFinalAv1(3) =
(IntAll1(3)+IntAll2(3)+IntAll3(3)+IntAll14(3)+IntAll5(3)+IntAll6(3)+IntAll7(3)+IntA
ll8(3)+IntAll9(3)+IntAll10(3)+IntAll11(3)+IntAll12(3)+IntAll13(3)+IntAll14(3)+IntAl
l15(3)+IntAll16(3)+IntAll17(3)+IntAll18(3)+IntAll19(3)+IntAll20(3))/19 % Manual
average due to NaN in IntAll10
TotalRetreat1 = IntFinalAv1(1) - IntFinalAv1(20); % Total retreat in displacement
(averaged)
Advance1 = IntFinal19(1) - IntFinal19(20);

%%%%% MAX Ur %%%%
UrMax = (IntFinal9(1) - IntFinal19(20))* (365/122) % Where 122 is the days elapsed
for Batch One
%%%%% AVERAGE Ur %%%%%%%
UrAv = TotalRetreat1 * 365/122
%%%%% Min Ur %%%%%%
UrMin = (IntFinal20(1) - IntFinal20(20))* 365/122
%%%% Error
ErrorUr = (-15.2-15.78)* 365/122

% ------- STATISTICAL ANALYSIS -----------
STD1 = std(IntFinalAv1

%%% BATCH TWO %%%%%%%%%%%%%%%%%%%%%%%%%%%%
IntFinalAv2 = IntAllAverage(21:45);
TotalRetreat2 = IntFinalAv2(1) - IntFinalAv2(end)
UrAv2 = TotalRetreat2 * 365/121
MaxUr2 = (IntFinalB5(1) - IntFinalB5(end)) * 365/121
MinUr2 = (IntFinalB19(1) - IntFinalB19(end)) * 365/121
ErrorUr2 = (-12.3-0.9) *365/121 % 39.81


%%% BATCH THREE %%%%%%%%%%%%%%%%%%%%%%%%%%%%
IntFinalAv3 = IntAllAverage(46:130);
%%% BATCH FOUR %%%%%%%%%%%%%%%%%%%%%%%%%%%%
IntFinalAv4 = IntAllAverage(131:end);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (7) CALCULATE THE ERROR BARS for each of the 156 points
% [A] Lake level errors from Date_FirstBatch, Date_LastBatch etc.
% BACK-PROJECTION ERROR (aka)

for i = 1:Size1(1)
LakeEr1x(i) = Date_FirstBatch(i).x_error;
LakeEr1y(i) = Date_FirstBatch(i).y_error;
LakeAbs1(i) = sqrt(LakeEr1x(i)^2+LakeEr1y(i)^2);
LakeEr1z(i) = Date_FirstBatch(i).z_error;
end

for i = 1:Size2(1)
```

```
LakeEr2x(i) = Date_Pre20th(i).x_error;
LakeEr2y(i) = Date_Pre20th(i).y_error;
LakeAbs2(i) = sqrt(LakeEr2x(i)^2+LakeEr2y(i)^2);
LakeEr2z(i) = Date_Pre20th(i).z_error;
end

for i = 1:Size3(1)
LakeEr3x(i) = Date_Post20th(i).x_error;
LakeEr3y(i) = Date_Post20th(i).y_error;
LakeAbs3(i) = sqrt(LakeEr3x(i)^2+LakeEr3y(i)^2);
LakeEr3z(i) = Date_Post20th(i).z_error;
end

for i = 1:Size4(1)
LakeEr4x(i) = Date_LastBatch(i).x_error;
LakeEr4y(i) = Date_LastBatch(i).y_error;
LakeAbs4(i) = sqrt(LakeEr4x(i)^2+LakeEr4y(i)^2);
LakeEr4z(i) = Date_LastBatch(i).z_error;
end

AllLakeErX = [LakeEr1x, LakeEr2x, LakeEr3x, LakeEr4x];
AllLakeErY = [LakeEr1y, LakeEr2y, LakeEr3y, LakeEr4y];
AllLakeAbs = [LakeAbs1, LakeAbs2, LakeAbs3, LakeAbs4];
AllLakeAbs = AllLakeAbs (1:end-2);

AllLakeErZ = [LakeEr1z, LakeEr2z, LakeEr3z, LakeEr4z];

% Plot Reprojection Errors as a Vector (test) (LAKE)
figure(15)
hold on
plot(DateAll1, AllLakeErX, 'r')
plot(DateAll1, AllLakeErY, 'g')
 % Hypotenuse of the X and Y errors
plot(DateAll1, AllLakeAbs, 'b')
hold off
datetick ('x',  'dd-mmm-yyyy');
title ('Tasman Terminus Reprojection Errors - X, Y and Total Reprojection Errors')
xlabel('Month and Year')
ylabel('Meters of Error (m)')
legend('X Error', 'Y Error', 'Total Reprojection Error')

% Plot Errors from GCP Trimble - Ordered from 1-20 GCP - GET REMAINDER OF
% GCP POINTS

TrimbleX = [0.1202, 0.244, 0.2645, 0.3311, 0.2596, 0.2934, 0.2757, 0.509, 0.2132,
0.3336, 0.3523, 0.3357, 0.3287, 0.0899 ]
TrimbleY = [0.0217, 0.1092, 0.1472, 0.2127, 0.1657, 0.17, 0.1217, 0.1118, 0.0538,
0.207, 0.1846, 0.2076, 0.2179, 0.0557]



% [B] RMS Error from the GCPs
% RMS error is a measure of where the GCPs are reprojected from...
% All roughly < 15m error, however depends on the location of the pixel in
% the image. Therefore it may be safe to average the RMS error and use this
% as pixels furthest from the camera have the highest RMS error while those
% closest are lowest <1m error.
% Batch One Average - 18 errors

B1ErX = (9.17 + 1.71 + 13.55+  12.84+ 4.9+ 11.23+ 9.89+ 11.15+ 12.59+ 4.81+ 1.51+
11.59+ 12.98+ 13.38+ 11.68+ 11.47+ 7.71+ 8.33)/18; %9.47
B1ErY = (1.97 + 0.39+ 4.91+ 4.57+ 1.71+ 3.91+ 3.28+ 2.41+ 2.76+ 1.07+ 0.31+ 2.34+
9.18+ 9.47+ 7.65+ 7.40 + 3.92+ 4.72)/ 18; %3.99
B1Av = sqrt(B1ErX^2 + B1ErY^2); % 10.28
for i = 1:Size1(1)
    B1Av(i) = sqrt(B1ErX^2 + B1ErY^2);
end

% Batch Two Average - 20 errors
```

```
B2ErX = (7.16+ 4.49+ 6.40+ 8.56+ 10.34+ 8.94+ 7.86+ 4.41 + 10.41+ 16.59+ 9.72+
6.98+ 11.66+ 15.46+ 3.62+ 3.44+ 4.94+ 6.39+ 14.40+ 7.08)/20; % 8.44
B2ErY = (0.97 + 0.64 + 1.17 + 1.05 + 3.82 + 3.27+ 3.74 + 2.13+ 1.42 + 2.30 + 1.37 +
0.92 + 1.49+ 2.01 + 1.62+ 1.54+ 2.05 + 2.61 + 4.63 + 2.54)/20; % 2.06
B2Av = sqrt(B2ErX^2 + B2ErY^2); % 8.6913
for i = 1:Size2(1)
    B2Av(i) = sqrt(B2ErX^2 + B2ErY^2);
end

% Batch Three Average - 11 new GCPs
B3ErX = (1.74 + 8.77+ 4.74+ 11.9+ 8.17+ 13.04+ 8.51+ 10.27+ 13.14+ 9.7+ 0.76)/ 11;
% 8.24
B3ErY = (0.24+ 1.21+ 0.67+ 1.57+ 1.04+ 5.83+ 3.81+ 4.25+ 5.36+ 3.12+ 0.27)/ 11; %
2.448
B3Av = sqrt(B3ErX^2 + B3ErY^2); % 8.6162
for i = 1:Size3(1)
    B3Av(i) = sqrt(B3ErX^2 + B3ErY^2);
end

% Batch Four Average - 15 errors

B4ErX = ( 12.83 + 4.81+ 11.18+ 23.11+ 6.18+ 12.40+ 5.82+ 12.09+ 8.15+ 12.86+ 14.54+
13.01+ 8.41+ 14.82+ 15.45)/15; %11.71
B4ErY = (1.75 + 0.69+ 2.05+ 2.83+ 0.84+ 1.72+ 0.82+ 1.59+ 1.04+ 5.75+ 6.51+ 5.39+
3.43+ 4.76+ 5.54)/15; % 2.98
B4Av = sqrt(B4ErX^2 + B4ErY^2); % 12.0840
for i = 1:26
    B4Av(i) = sqrt(B4ErX^2 + B4ErY^2);
end

% Total in a vector

AllGCPEr = [B1Av, B2Av, B3Av, B4Av];
AllGCPEr = AllGCPEr ([1:48 50:end-2]);
% [B2] Inherent GCP error from the Trimble GNSS
% Trimble R8 http://trl.trimble.com/docushare/dsweb/Get/Document-140079/022543-
079N_TrimbleR8GNSS_DS_1014_LR.pdf

TrimbleErY = (0.207+0.1846+0.2076+ 0.2179+ 0.0557 + 0.0217+ 0.1092 + 0.1472 +
0.2127+ 0.1657 + 0.17 + 0.1217 + 0.1118 + 0.0539) / 14; % Longitude 0.1419m
TrimbleErX = (0.3336 + 0.3523 + 0.3357 + 0.3287 + 0.0899 + 0.1202 + 0.244+ 0.2645+
0.3311 + 0.2596 + 0.2934 + 0.2757 + 0.509 + 0.2132)/14; % Latitude 0.2822m

TrimbleAv = sqrt(TrimbleErY^2 + TrimbleErX^2);
for i = 1:153 % Create a vector for each point = TrimbleAv constant
    TrimbleAv(i) = sqrt(TrimbleErY^2 + TrimbleErX^2);
end

% [C] Marginal Camera movement error
% Difficult to estimation? ASSUME TO BE 0 after post processing


% [D] Human error in digitizations
% Assume to be roughly +/- 2 pixels from manual digitizations
for i = 1:153
    HumanErAll(i) = 0.5; % Constant error
end

% [E] FINAL ERROR
FinalErrorAv = AllGCPEr + AllLakeAbs + TrimbleAv + HumanErAll; % Averaged and at
every point

% Brian's polygonal solution
ppf = patch([DateAll1; flipud(DateAll1)],...
    [min(AllLakeErX,[],2); AllLakeErX(max(AllLakeErX,[],2))],'r',...
    'facecolor',[1,0.5,0.5],'edgecolor','r')'%,'facealpha',0.7)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%
```

```
% SUM ALL RESULTS FROM TRANSECTS for PLOTTING AVERAGE ERROR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure (16)
hold on
s = plot(DateAll1, IntAll1,'Color', cmap(1,:));
set(get(get(s,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); % Turns
off its entry in Legend
%alpha(s,.5) % Change Transparency levels
s2 = plot(DateAll1, IntAll2,'Color', cmap(2,:));
set(get(get(s2,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s3 = plot(DateAll1, IntAll3,'Color', cmap(3,:));
set(get(get(s3,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s4 = plot(DateAll1, IntAll4,'Color', cmap(4,:))
set(get(get(s4,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s5 = plot(DateAll1, IntAll5,'Color', cmap(5,:))
set(get(get(s5,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s6 = plot(DateAll1, IntAll6,'Color', cmap(6,:))
set(get(get(s6,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s7 = plot(DateAll1, IntAll7,'Color', cmap(7,:))
set(get(get(s7,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s8 = plot(DateAll1, IntAll8,'Color', cmap(8,:))
set(get(get(s8,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s9 = plot(DateAll1, IntAll9,'Color', cmap(9,:))
set(get(get(s9,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s10 = plot(DateAll1, IntAll10,'Color', cmap(10,:))
set(get(get(s10,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s11 = plot(DateAll1, IntAll11,'Color', cmap(11,:))
set(get(get(s11,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s12 = plot(DateAll1, IntAll12,'Color', cmap(12,:))
set(get(get(s12,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s13 = plot(DateAll1, IntAll13,'Color', cmap(13,:))
set(get(get(s13,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s14 = plot(DateAll1, IntAll14,'Color', cmap(14,:))
set(get(get(s14,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s15 = plot(DateAll1, IntAll15,'Color', cmap(15,:)) % Almost identical to transect 1
set(get(get(s15,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s16 = plot(DateAll1, IntAll16,'Color', cmap(16,:))
set(get(get(s16,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s17 = plot(DateAll1, IntAll17,'Color', cmap(17,:))
set(get(get(s17,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s18 = plot(DateAll1, IntAll18,'Color', cmap(18,:))
set(get(get(s18,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s19 = plot(DateAll1, IntAll19,'Color', cmap(19,:))
set(get(get(s19,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend
s20 = plot(DateAll1, IntAll20,'Color', cmap(20,:))
set(get(get(s20,'Annotation'),'LegendInformation'),'IconDisplayStyle','off'); %
Turns off its entry in Legend

dateV = DateAll1;
```

```
h1 = plot(dateV, IntAllAverage, 'k');
h1.LineWidth = 2.5; % Change the width of the average plot
plot(DateAll1, AllGCPEr, 'yv')
plot(DateAll1, AllLakeAbs, 'bv')
plot(DateAll1, TrimbleAv, 'cv')
plot(DateAll1, HumanErAll, 'v','Color', [1, 0.5, 0])
h2 = errorbar(DateAll1, IntAllAverage, FinalErrorAv, 'k');
h2.LineWidth = 0.38;
%plot(dateV, IntAllAverage, 'g');
hold off
%h.LineStyle = ':';

hax = get(h1, 'Parent'); % gets the parent axes handle;
set(hax, 'XTick', [dateV(1:30:end)]);
datetick ('x',  'dd-mmm-yyyy', 'keepticks');
title ('Tasman Terminus Average Positions (2013-2016) with Total Errors')
xlabel('Month and Year')
ylabel('Meters (m)')
axis tight
legend('Average Terminus Positions 2013-2016', 'Camera Matrix Errors', 'Lake
Reprojection Errors', 'GCP Acquisition Errors', 'Human Errors', 'Total Data
Acquisition Errors')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CALCULATING ANNUAL Ur / positional changes
%%%% 2013-2014 - 5th February 2013 to 5th February 2014
annualUr1 = IntFinalAv1(1) - IntFinalAv3(12)
ErrorYearUr1 = (YearError1+14) % 17.65

%%%% 2014-2015
annualUr2= IntFinalAv3(12) - IntFinalAv3(49)

%%%% 2015-2016
annualUr3 = IntFinalAv3(49) - IntFinalAv4(end)

%%%% ALL BATCHES %%%%
finalUr = (IntFinalAv1(1) - IntFinalAv4(end))* 365/1141
%%% Minimum position change
finalUrmin = (IntFinal20(1) - IntFinalD20(end))* 365/1141

%%% Maximum position change
finalUrmin = (IntFinal3(1) - IntFinalD3(end))* 365/1141



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLOT ALL ERRORS ONLY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure (17)
hold on
% AllGCPEr = AllGCPEr([1:48 50:end]);
% AllLakeAbs = AllLakeAbs([1:48 50:end]);
% TrimbleAv = TrimbleAv([1:48 50:end]);
% HumanErAll = HumanErAll([1:48 50:end]);
% FinalErrorAv = FinalErrorAv([1:48 50:end]);

plot(DateAll1, AllGCPEr, 'g')
plot(DateAll1, AllLakeAbs, 'b')
plot(DateAll1, TrimbleAv, 'c')
plot(DateAll1, HumanErAll, 'y')
h3 = plot(DateAll1, FinalErrorAv, 'r')
h3.LineWidth = 1;
hold off
%hax = get(h1, 'Parent'); % gets the parent axes handle;
%set(hax, 'XTick', [dateV(1:30:end)]);
datetick ('x',  'dd-mmm-yyyy', 'keepticks');
title ('Total Error Constituents', 'Fontsize', 16)
xlabel('Month and Year','Fontsize', 16)
```

```
ylabel('Meters of Error (m)','Fontsize', 16)
legend( 'Camera Matrix Errors', 'Lake Reprojection Errors', 'GCP Acquisition
Errors', 'Human Errors', 'Total Error')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLOT AVERAGE POSITION AGAINST TEMPERATURE/PRECIPITATION - Why did 2014 have no
massive
% calving events?
% Was it temperature related???
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define some colours - light blue
RGB1 = [132, 112, 225]; % Light Slate Blue
MATC1 = RGB1/255;
% Orange
RGB2 = [255, 153, 51];
MATC2 = RGB2/255;
% Dark Red
RGB3 = [204, 0, 0];
MATC3 = RGB3/255;
% Nice Blue
RGB4 = [0, 128, 255];
MATC4 = RGB4/255;


% -----------------------------------
% PRECIPITATION v POSITION (data from Plot_Rainfall_Eds.m)
% -----------------------------------

figure(18)
ylim([DateAll1(1) ,DateAll1(end)])
yyaxis left % define left axis
plot(DateAll1, IntAllAverage, 'Color', MATC2);
datetick ('x',  'dd-mmm-yyyy', 'keepticks');
title ('Average Terminus Positions v Cumulative Daily Precipitation (Mt.Cook EWS)')
xlabel('Month and Year')
ylabel('Meters (m)')

yyaxis right % define right axis plot
plot (t, RainDailyCum, 'Color', MATC1);
ylabel ('mm/day')
legend( 'Average Terminus Position','Mt.Cook EWS (mm/day)')

% -----------------------------------------------
% PLOT FOR EXTREME MAX MONTHLY AIR TEMPERATURE
% MT Cook EWS 18125
% -----------------------------------------------

figure(19)
ylim([DateAll1(1) ,DateAll1(end)])
yyaxis left % define left axis
plot(DateAll1, IntAllAverage, 'Color', MATC2);
datetick ('x',  'dd-mmm-yyyy', 'keepticks');
title ('Average Terminus Positions v Extreme Maximum Monthly Air Temperature
(Mt.Cook EWS)')
xlabel('Month and Year')
ylabel('Meters (m)')

% Define Air Extreme Temp
% 2013: Feb-Dec, 2014: Jan-Dec, 2015: Jan-Dec, 2016: Jan-March
AirExtremeMax = [30.8,27.4,21.0,NaN,12.0,16.5,14.7,18.2,22.2,23.9,26.4,
27.0,29.8,26.2,24.4,18.7,14.3,13.4,15.5,22.7,23.6,23.2,26.8,
29.8,25.8,25.0,23.8,20.1,14.8,16.8,15.0,17.4,22.3,23.5,27.7,  26.3, 28.7, 24.2]; %
Feb 2013 to March 2016
t1 = datetime(2013,2,1,0,0,0); % 1st Feb 2013
t2 = datetime(2016,3,30,0,0,0); % 30th March 2016
AirExtremeTime = t1:calmonths(1):t2; % using calmonths function, it will iterate
based on a month. Caldays. Calmonths. HOW COOL!
yyaxis right % define right axis plot
```

```
plot (AirExtremeTime, AirExtremeMax, 'Color', MATC3);
ylabel ('deg C')
legend( 'Average Terminus Position','Mt.Cook EWS Extreme Max Temp (deg C)')

% ------------------------------------------------
% Average Daily Temperature 2013-2016 v Position
% ------------------------------------------------
figure(20)
subplot(2,2,1)
ylim([DateAll1(1) ,DateAll1(end)])
yyaxis left % define left axis
plot(DateAll1, IntAllAverage);
datetick ('x',  'dd-mmm-yyyy', 'keepticks');
title ('Average Terminus Positions v Daily Max Air Temperature (Mt.Cook EWS)')
xlabel('Month and Year')
ylabel('Meters (m)')
axis tight

% Define Average Temp
% Full Time duration 2013-2016 - STORED ON DESKTOP in NOTEPAD
% Manually Import the data % FullTMax, FullTMin, DailyTDates --- do this

% Concatenate the dates
DailyTNum = cellfun(@(x) x(1:8),DailyTDates(cellfun('length',DailyTDates) >
7),'un',0);
DailyTNumDates = datenum(DailyTNum, 'yyyymmdd');

%t1 = datetime(2013,2,1,0,0,0); % 1st Feb 2013
%t2 = datetime(2016,3,30,0,0,0); % 30th March 2016
%AverageTempTime = t1:caldays(1):t2; % using calmonths function, it will iterate
based on a month. Caldays. Calmonths. HOW COOL!
yyaxis right % define right axis plot
hold on
plot (DailyTNumDates, FullTMax, ':', 'Color', MATC3);
plot (DailyTNumDates, FullTMin, ':', 'Color', MATC4);
hold off
ylabel ('deg C')
legend( 'Average Terminus Position','Mt.Cook EWS Daily Max Air Temp (deg
C)','Mt.Cook EWS Daily Min Air Temp (deg C)' )

%%%%%% ATTAIN TEMPERATURE (Min and Max) at each timestep
%%% Using a logical Index from the Intersection function %%%%

[TC, TIA, TIB] = intersect(DateAll1, DailyTNumDates);
FullTMaxDiscrete = FullTMax(TIB); % Solves the problem of attaining Temp at each
timestep
FullTMinDiscrete = FullTMin(TIB);
DailyNumDatesInt = DailyTNumDates(TIB);
%%%%% AVERAGE THE TEMPERATURE DATA / SMOOTH or FILTER IT SO WE ARE NOT
%%%%% DEALING WITH JUST INSTANTANEOUS DATA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



%%% PROBLEM OF SIZES NOT MATCHING. e.g. IntAllAverage = 153 entries,
%%% whereas FullTMax has only 151 entries. Upon looking it up using
%%% --- ENTRY 49 in INTALLAVERAGE is MISSING ---
%%% --- ENTRY 153 is Also missing ---
%IntAllAverageNew = IntAllAverage([1:48, 50:end-1]) % Removes entry 49 and last
entry
%DateAll1New = DateAll1([1:48, 50:end-1])

%%%%%%%%%%% LINEAR REGRESSION PLOT %%%%%%%%%%%%%%%%
%%%% Position vs Temperature R^2 %%%%%%%%%%%%%%%%%%
IntAllAverageNewT = transpose(IntAllAverageNew); % Can't do regression with row
vectors?
FullTMaxDiscreteT = transpose(FullTMaxDiscrete);
bT1 = IntAllAverageNewT\FullTMaxDiscreteT % Calculate the slope for linear/least
squares regression
```

```
%%%% Correlation Coefficient %%%%h
RT1 = corrcoef(IntAllAverageNew, FullTMaxDiscrete)


%%% Try plotting it against each other
figure (1)
hold on
yyaxis left
plot(DateAll1New, IntAllAverageNew, 'LineWidth', 2)
plot(DateAll1New, IntAllAverageNew,'x', 'LineWidth', 2) % Plot the Intersections
first
ylabel('Meters (m)', 'Fontsize', 16)

yyaxis right
plot(DailyNumDatesInt, FullTMaxDiscreteT, 'x', 'LineWidth', 2)
plot(DailyNumDatesInt, FullTMaxDiscreteT, ':', 'LineWidth', 2) % Plot only the
intersected Temperature gradients
title ('Average Terminus Positions v Daily Max Air Temperature Averaged +/- 5 days
of each sample (Mt.Cook EWS)','Fontsize', 18)
xlabel('Month and Year','Fontsize', 16)
ylabel('Degrees Celcius', 'Fontsize', 16)
legend('Average Terminus Position', '5-10 days per measurement', 'Daily Max
Temperature', 'Average of +/- 5 days of date where terminus positions are
available')
axis image

datetick ('x', 'mmmyyyy')


figure (2)
plot(IntAllAverageNewT, FullTMaxDiscreteT)



% ------------------------------------------------------------------------
% Average Daily Temperature 2013-2016 v Position (SMOOTHED to WEEKLY)
% ------------------------------------------------------------------------









% ------------------------------------------------------------------------
% lake Levels 2013-2016 v Position (SMOOTHED to WEEKLY)
% ------------------------------------------------------------------------

% Lake levels
% Sampled hourly from 1st February 2012 - 19th March 2016.
% MANUAL IMPORTATION - LakeDate, LakeLevel, LakeError

% Average Lake Level and Average Dates % DIFFICULT TO SCRIPT % Average of
% every n entries in a vector Help!
LakeDateDay = cellfun(@(x) x(1:10),LakeDate(cellfun('length',LakeDate) >
7),'un',0);
LakeDatenum = datenum(LakeDate, 'yyyy-MM-dd hh:mm:ss');

% (A) Using Arrayfun to solve this problem
%n = 24; % average every n values
%a = reshape(cumsum(ones(n,10),2),[],1); % Arbitrary data
%LakeLevelDay = LakeLevel(@(i) mean(a(i:i+n-1)),1:n:length(a)-n+1)'; % the averaged
vector

% (B) Using BLOCK CONCEPT To average it every 24 hours to attain daily amounts
% IMPORT LAKELEVELS %PulseRateF = rand(399277, 1);
```

```
% Define the block parameter.  Average in a 100 row by 1 column wide window.
blockSize = [24, 1];
% Block process the image to replace every element in the
% 24 element wide block by the mean of the pixels in the block.
% First, define the averaging function for use by blockproc().
meanFilterFunction = @(theBlockStructure) mean2(theBlockStructure.data(:));
% Now do the actual averaging (block average down to smaller size array).
LakeBlock = blockproc(LakeLevel, blockSize, meanFilterFunction);
% Let's check the output size.
[rows, columns] = size(LakeBlock);

% Convert LakeDate to a 1509x1 matrix
LakeDateDay = LakeDateDay(1:24:end);
%LakeDatenumday = datenum(LakeDateDay, 'yyyy-MM-dd'); % Convert to datenum
% SOLUTION
for i = 1:length(LakeDateDay);
    datecount = LakeDateDay(i); % Need to convert the DATE to a STRING first using
datestr
   LakeDatenumday(i) = datenum(cellfun(@(x) {[x(1:4) '-' x(6:7) '-' x(9:10)]},
LakeDateDay(i)), 29); % Convert Datetime to Datenum because errorbar function
doesn't like datetime formats
end
% Error checking Datenum - once again a problem with the Datenum!!!
datestr(LakeDatenumday) % PERFECT after SOLUTION

% Initial Plot
figure(20)
ylim([DateAll1(1) ,DateAll1(end)])
yyaxis left % define left axis
plot(DateAll1, IntAllAverage, 'Color', 'k', 'MarkerSize', 10 );
datetick ('x',  'dd-mmm-yyyy', 'keepticks');
title ('Average Terminus Positions v Lake Levels')
xlabel('Month and Year')
ylabel('Meters (m)')
yyaxis right % define right axis plot
hold on
plot (LakeDatenumday, LakeBlock, 'Color', MATC2);
hold off
ylabel ('Meters (m)')
legend( 'Average Terminus Position','Tasman Lake Levels (m)')

% --------------------------------
% LAKE LEVELS CLOSER INSPECTION
% --------------------------------
% Running standard deviation of lake level v Rate of Change of Terminus
% Position

% Retreat Rate
% The rate at which
IntAllAverage(88) = 0
RetreatRateAv = trapz(DateAll1, IntAllAverage)

for i = 1:length(IntAllAverage)-1
    RetreatRateRunning(i) = trapz(DateAll1(i:i+1), IntAllAverage(i:i+1));
end
figure(20)
plot(DateAll1(2:end), RetreatRateRunning)

datetick ('x',  'dd-mmm-yyyy', 'keepticks');
title ('Retreat Rates')
xlabel('Month and Year')
ylabel('Retreat Rate (dm/dt)')

% LAKE RATES OF CHANGE
% Weekly Mean

blockSize = [7, 1];
meanFilterFunction = @(theBlockStructure) mean2(theBlockStructure.data(:));
% Now do the actual averaging (block average down to smaller size array).
```

```
LakeBlockWeekly = blockproc(LakeBlock, blockSize, meanFilterFunction);
% Let's check the output size.
[rows, columns] = size(LakeBlockWeekly);

% Monthly Mean
blockSize = [7, 1];



figure (21)
hold on
plot (LakeDatenumday(1:7:end), LakeBlockWeekly, 'Color', MATC3);
plot (LakeDatenumday, LakeBlock, 'Color', MATC2);
plot(DateAll1(2:end), RetreatRateRunning)

datetick ('x',  'dd-mmm-yyyy', 'keepticks');
title ('Retreat Rates')
xlabel('Month and Year')
ylabel('Retreat Rate (dm/dt)')
hold off



% ------------------------------------------------------------------------------
-----------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
% (21) ESTIMATING AREA FROM INTERSECTIONS
% Idea - was to use Polyarea function
% OTHER IDEA - create a mask, then binary the image, then use bwarea

% Create horizontal boundary transect using Ginput

figure(21)
%ax.XLimMode = 'manual'; % Changes Xlimit Mode to Manual rather than Auto
%ax.YLimMode = 'manual';
%xlim([5.1614e6, 5.1623e6])
%ylim([1.3726e6, 1.374e6])
plot([FirstBatch.X], [FirstBatch.Y], 'k')
hold on
plot([Pre20thBatch.X], [Pre20thBatch.Y], 'k')
plot([Post20thBatch.X], [Post20thBatch.Y], 'k')
plot([LastBatch.X], [LastBatch.Y], 'k')
plot(cell2mat(TransectXFinal(1)), cell2mat(TransectYFinal(1)), 'Color', cmap(1,:))
plot(cell2mat(TransectXFinal(2)), cell2mat(TransectYFinal(2)), 'Color', cmap(2,:))
plot(cell2mat(TransectXFinal(3)), cell2mat(TransectYFinal(3)), 'Color', cmap(3,:))
plot(cell2mat(TransectXFinal(4)), cell2mat(TransectYFinal(4)), 'Color', cmap(4,:))
plot(cell2mat(TransectXFinal(5)), cell2mat(TransectYFinal(5)), 'Color', cmap(5,:))
plot(cell2mat(TransectXFinal(6)), cell2mat(TransectYFinal(6)), 'Color', cmap(6,:))
plot(cell2mat(TransectXFinal(7)), cell2mat(TransectYFinal(7)), 'Color', cmap(7,:))
plot(cell2mat(TransectXFinal(8)), cell2mat(TransectYFinal(8)), 'Color', cmap(8,:))
plot(cell2mat(TransectXFinal(9)), cell2mat(TransectYFinal(9)), 'Color', cmap(9,:))
plot(cell2mat(TransectXFinal(10)), cell2mat(TransectYFinal(10)), 'Color',
cmap(10,:))
plot(cell2mat(TransectXFinal(11)), cell2mat(TransectYFinal(11)), 'Color',
cmap(11,:))
plot(cell2mat(TransectXFinal(12)), cell2mat(TransectYFinal(12)), 'Color',
cmap(12,:))
plot(cell2mat(TransectXFinal(13)), cell2mat(TransectYFinal(13)), 'Color',
cmap(13,:))
plot(cell2mat(TransectXFinal(14)), cell2mat(TransectYFinal(14)), 'Color',
cmap(14,:))
plot(cell2mat(TransectXFinal(15)), cell2mat(TransectYFinal(15)), 'Color',
cmap(15,:))
plot(cell2mat(TransectXFinal(16)), cell2mat(TransectYFinal(16)), 'Color',
cmap(16,:))
```

```
plot(cell2mat(TransectXFinal(17)), cell2mat(TransectYFinal(17)), 'Color',
cmap(17,:))
plot(cell2mat(TransectXFinal(18)), cell2mat(TransectYFinal(18)), 'Color',
cmap(18,:))
plot(cell2mat(TransectXFinal(19)), cell2mat(TransectYFinal(19)), 'Color',
cmap(19,:))
plot(cell2mat(TransectXFinal(20)), cell2mat(TransectYFinal(20)), 'Color',
cmap(20,:))
ax.XGrid = 'on'; % Toggle X Grid Lines
ax.YGrid = 'on'; % Y grid
axis manual % tight, fill, etc.
grid on
title('Tasman Glacier Terminus Positions February 2013 - March 2016 - Area Boundary
Transect')
xlabel ('X')
ylabel ('Y')

% (8) GUINPUT for Transect Boundary - DRAW THE TRANSECT!
% PLOT THE BOUNDARY ON THE BOTTTOM THAT INTERSECTS WITH ALL TRANSECTS
[Lx4,Ly4]= ginput(1);
[Lx5, Ly5] = ginput(1); % Ginput asking for how many clicks. 1.
A4 = [Lx4, Lx5]; % Need to be in this format so as not to plot a line that FITS the
points, rather a line that only fits the two points
% Get the Bottom location of transect clicked
A5 = [Ly4, Ly5];
scatter (A4, A5);
plot(A4, A5, 'r');
X = [Lx4, Ly4; Lx5, Ly5];
Aslope = (Ly5 - Ly4) ./ (Lx5 - Lx4);
str2 = num2str(Aslope); % m = tan(theta) is the conversion to an angle from slopes
text(Lx4, Ly4, str2);

% Create a transect similar to the other Transects
% Then find intersections with the Boundary Transect number 1 and number 20
% i.e TransectXFinal1, TransectYFinal1, TransectXFinal20, TransectYFinal20
AreaT1X4 = AreaT1X4(3:end); % Due to overlap
AreaT1Y4 = AreaT1Y4(3:end); % Due to overlap
AreaT20X4 = AreaT20X4(3:end); % Due to overlap
AreaT20Y4 = AreaT20Y4(3:end); % Due to overlap
% Aggregate
AreaT1FinalX = [AreaT1X1, AreaT1X2, AreaT1X3, AreaT1X4];
AreaT20FinalX = [AreaT20X1, AreaT20X2, AreaT20X3, AreaT20X4];
AreaT1FinalY = [AreaT1Y1, AreaT1Y2, AreaT1Y3, AreaT1Y4];
AreaT20FinalY = [AreaT20Y1, AreaT20Y2, AreaT20Y3, AreaT20Y4];
%scatter(AreaT1FinalX, AreaT1FinalY, 'b')
%scatter(AreaT20FinalX, AreaT20FinalY, 'r')
% Find intersection of Boundary Transect and T1 and T20
BoundaryLineX = linspace(Lx4, Lx5, 142); % The number of points is arbitrary but
since length(TransectXFinal1(1)) = 142. We use 142.
BoundaryLineY = linspace(Ly4, Ly5, 142);
% T1
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal1(1)),
cell2mat(TransectYFinal1(1)), BoundaryLineX, BoundaryLineY, 1); %It works
BoundaryX1 = x1;
BoundaryY1 = y1;
%%%%%% Calculating intersection between Transect 20 and each TERMINUS -
%%%%%% done already using AreaT1X1 etc.



%%%%%%%
% T20
[x1, y1, iout1, jout1] = intersections(cell2mat(TransectXFinal20(1)),
cell2mat(TransectYFinal20(1)), BoundaryLineX, BoundaryLineY, 1); %It works
BoundaryX20 = x1;
BoundaryY20 = y1;
scatter(BoundaryX1, BoundaryY1, 'c');
scatter(BoundaryX20, BoundaryY20, 'c');
```

```
hold off

% FINAL VERTICES ARE
FinalAreaVertexX = [BoundaryX1, BoundaryX20, AreaT20FinalX(end),
AreaT1FinalX(end)];
FinalAreaVertexY = [BoundaryY1, BoundaryY20, AreaT20FinalY(end),
AreaT1FinalY(end)];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CALCULATE THE AREA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% % Create similarly spaced Boundary Transect to terminus digitizations
% for i = 1:Size1(1);
% BatchX = Post20thBatch(i).X;
% VectorSize = size(BatchX);
% VectorScl(i) = VectorSize(2);
% end
% for k = 1:Size1(1)
% TransectX = linspace(BoundaryX1, BoundaryX20, VectorScl(k)); % Where 371 is the
number of vector points for FirstBatch(1).X*-
% TransectY = linspace(BoundaryY1, BoundaryY20, VectorScl(k));
% BoundaryTransectsX1{k} = TransectX; % WHAT IF I DONT STORE IT
% BoundaryTransectsY1{k} = TransectY;
% end
% for i = 1:Size2(1);
% BatchX = Post20thBatch(i).X;
% VectorSize = size(BatchX);
% VectorScl(i) = VectorSize(2);
% end
% for k = 1:Size2(1);
% TransectX = linspace(BoundaryX1, BoundaryX20, VectorScl(k));
% TransectY = linspace(BoundaryY1, BoundaryY20, VectorScl(k));
% BoundaryTransectsX2{k} = TransectX; % WHAT IF I DONT STORE IT
% BoundaryTransectsY2{k} = TransectY;
% end
% for i = 1:Size3(1);
% BatchX = Post20thBatch(i).X;
% VectorSize = size(BatchX);
% VectorScl(i) = VectorSize(2);
% end
% for k = 1:Size3(1)
% TransectX = linspace(BoundaryX1, BoundaryX20, VectorScl(k));
% TransectY = linspace(BoundaryY1, BoundaryY20, VectorScl(k));
% BoundaryTransectsX3{k} = TransectX; % WHAT IF I DONT STORE IT
% BoundaryTransectsY3{k} = TransectY;
% end
% for i = 1:Size4(1);
% BatchX = Post20thBatch(i).X;
% VectorSize = size(BatchX);
% VectorScl(i) = VectorSize(2);
% end
% for k = 1:Size4(1)
% TransectX = linspace(BoundaryX1, BoundaryX20, VectorScl(k));
% TransectY = linspace(BoundaryY1, BoundaryY20, VectorScl(k));
% BoundaryTransectsX4{k} = TransectX; % WHAT IF I DONT STORE IT
% BoundaryTransectsY4{k} = TransectY;
% end
% BoundaryTransectsXFinal = [BoundaryTransectsX1, BoundaryTransectsX2,
BoundaryTransectsX3, BoundaryTransectsX4]
% BoundaryTransectsYFinal = [BoundaryTransectsY1, BoundaryTransectsY2,
BoundaryTransectsY3, BoundaryTransectsY4]
%


% Create evenly spaced Transect Points

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Loop for every Area calculation
%%

BoundaryLineX = linspace(Lx4, Lx5, 142); % The number of points is arbitrary but
since length(TransectXFinal1(1)) = 142. We use 142.
BoundaryLineY = linspace(Ly4, Ly5, 142);
% for i = 1:154; % Size of transect 1 and 20 Intersections with Termini
% TransectBoundX{i} = linspace(Lx4, AreaT1FinalX(i),142);
% TransectBoundY{i} = linspace(Ly4, AreaT1FinalY(i),142);
% end


%%% PROBLEM 1: Ascending (Left to Right) and Descending (Right to Left) order of
vectors in FirstBatch.X
%%% and FirstBatch.Y was solved by figuring out whether it is descending or
%%% ascending in the X-axis. For those termini in the wrong direction, it would be
%%% flipped using fliplr.
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% SURFACE AREA %%%%%%%%%%%%%%%%
%%% BATCH ONE %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure (99)
[FirstBatch, Date_FirstBatch] = shaperead('UNPROJECT_6months_2d.shp'); % Use this
to reverse changes
for j = 1:20
    if FirstBatch(j).X(2)< FirstBatch(j).X(end-1)
       FirstBatch(j).X = fliplr(FirstBatch(j).X); % Sorts it based on descending
order
       FirstBatch(j).Y = fliplr(FirstBatch(j).Y); % Switches y to Left to Right
using fliplr command
    else
       FirstBatch(j).X = FirstBatch(j).X;
       FirstBatch(j).Y = FirstBatch(j).Y;
    end
end

% Batch 1
for i = 1:Size1(1); % Number of terminus in total
    clear resultX
    clear resultY
    ix =  AreaT20X1(i) >= FirstBatch(i).X | FirstBatch(i).X >= AreaT1X1(i); % This
makes a 1 for all those that satisfy this condition (FINAL CHECK)
    resultX = FirstBatch(i).X;
    resultX(ix) = NaN;
    resultY = FirstBatch(i).Y;
    resultY(ix) = NaN;
    % Remove all entries that are NaN - this was causing problems where the
    % lines did not join in polyarea.
    resultX(find(isnan(resultX))) = []
    resultY(find(isnan(resultY))) = []
    % Counter clockwise = + area , clockwise = - area , currently resultx =
    % Left to right - Vector creation here
    AllX = [ BoundaryX20, AreaT20X1(i), fliplr(resultX),  AreaT1X1(i), BoundaryX1,
BoundaryX20]; % Here i create a X vector with points that can be joined up to
ascertain an area
    AllY = [ BoundaryY20, AreaT20Y1(i), fliplr(resultY), AreaT1Y1(i), BoundaryY1,
BoundaryY20]; % Has The terminus, the vertices of the intersections
%     AllX = [ BoundaryX20, AreaT20X1(i), resultX, AreaT1X1(i), BoundaryX1,
BoundaryX20]; % Here i create a X vector with points that can be joined up to
ascertain an area
%     AllY = [ BoundaryY20, AreaT20Y1(i), resultY, AreaT1Y1(i), BoundaryY1,
BoundaryY20];
    hold on
    subplot(2,2,1)
    plot(AllX, AllY)
    %plot(AllXmax, AllYmax)
    text(Lx4-50, Ly4+30, str2); % slope angle in radians
```

```
    grid off
    title('Batch One Area Measurements', 'Fontsize',20)
    xlabel('NZTM X (meters)')
    ylabel('NZTM Y (meters)')
    hold off
    Area1(i) = polyarea(AllX, AllY);
    Area1
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% SURFACE AREA - PLANFORM PLOT %%%%%%%%%%%%%%%%%
%%% BATCH THREE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for j = 1:Size2(1);
    if Pre20thBatch(j).X(2)< Pre20thBatch(j).X(end-1)
       Pre20thBatch(j).X = fliplr(Pre20thBatch(j).X); % Sorts it based on
descending order
       Pre20thBatch(j).Y = fliplr(Pre20thBatch(j).Y); % Switches y to Left to Right
using fliplr command
    else
       Pre20thBatch(j).X = Pre20thBatch(j).X;
       Pre20thBatch(j).Y = Pre20thBatch(j).Y;
    end
end
% Batch 2
for i = 1:Size2(1); % Number of termini in total
    clear resultX
    clear resultY
    ix =  AreaT20X2(i) >= Pre20thBatch(i).X | Pre20thBatch(i).X >= AreaT1X2(i); %
This makes a 1 for all those that satisfy this condition (FINAL CHECK)
    resultX = Pre20thBatch(i).X;
    resultX(ix) = NaN;
    resultY = Pre20thBatch(i).Y;
    resultY(ix) = NaN;
    % Remove all entries that are NaN - this was causing problems where the
    % lines did not join in polyarea.
    resultX(find(isnan(resultX))) = []
    resultY(find(isnan(resultY))) = []
    % Counter clockwise = + area , clockwise = - area , currently resultx =
    % Left to right - Vector creation here
    AllX = [ BoundaryX20, AreaT20X2(i), fliplr(resultX),  AreaT1X2(i), BoundaryX1,
BoundaryX20]; % Here i create a X vector with points that can be joined up to
ascertain an area
    AllY = [ BoundaryY20, AreaT20Y2(i), fliplr(resultY), AreaT1Y2(i), BoundaryY1,
BoundaryY20]; % Has The terminus, the vertices of the intersections
%     AllX = [ BoundaryX20, AreaT20X1(i), resultX, AreaT1X1(i), BoundaryX1,
BoundaryX20]; % Here i create a X vector with points that can be joined up to
ascertain an area
%     AllY = [ BoundaryY20, AreaT20Y1(i), resultY, AreaT1Y1(i), BoundaryY1,
BoundaryY20];
    hold on
    subplot(2,2,2)
    plot(AllX, AllY)
    %plot(AllXmax, AllYmax)
    text(Lx4-50, Ly4+30, str2);
    grid off
    title('Batch Two Area Measurements','Fontsize',20)
    xlabel('NZTM X (meters)')
    ylabel('NZTM Y (meters)')
    hold off
    Area2(i) = polyarea(AllX, AllY);
    Area2
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% SURFACE AREA %%%%%%%%%%%%%%%
%%% BATCH THREE %%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
for j = 1:Size3(1);
    if Post20thBatch(j).X(2)< Post20thBatch(j).X(end-1)
       Post20thBatch(j).X = fliplr(Post20thBatch(j).X); % Sorts it based on
descending order
       Post20thBatch(j).Y = fliplr(Post20thBatch(j).Y); % Switches y to Left to
Right using fliplr command
    else
       Post20thBatch(j).X = Post20thBatch(j).X;
       Post20thBatch(j).Y = Post20thBatch(j).Y;
    end
end
% Batch 3
subplot(2,2,3)

hold on
for i = 1:12; % Number of termini in total
    clear resultX
    clear resultY
    clear ix
    ix =  AreaT20X3(i)  >= Post20thBatch(i).X | Post20thBatch(i).X >= AreaT1X3(i);
% This makes a 1 for all those that satisfy this condition (FINAL CHECK)
    % Calibration to remove errors
    if i == 1 %  This is because some values went backwards in X
        clear ix
        ix =  AreaT20X3(1)  >= Post20thBatch(1).X | Post20thBatch(1).X >=
AreaT1X3(1);
        ix(75:133) = 1;
        resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections

        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        title('Batch Three Area Measurements','Fontsize',20)
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);

    end
    if i == 2 % Cuts off backwards values for 2nd termini
       clear ix
        ix =  AreaT20X3(2)  >= Post20thBatch(2).X | Post20thBatch(2).X >=
AreaT1X3(2);
        ix(58:87) = 1; % Assigning 1 makes them NaN
        resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
```

```
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);

    end
    if i == 3 % Cuts off backwards values for 3rd termini
       clear ix
       ix =  AreaT20X3(3)  >= Post20thBatch(3).X | Post20thBatch(3).X >=
AreaT1X3(3);
       ix(68:114) = 1;
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);

    end
    if i == 4 % Cuts off backwards values for 4th termini
       clear ix
       ix =  AreaT20X3(4)  >= Post20thBatch(4).X | Post20thBatch(4).X >=
AreaT1X3(4);
       ix(44:72) = 1;
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);

    end
    if i == 5 % Cuts off backwards values for 5th termini
       clear ix
       ix =  AreaT20X3(5)  >= Post20thBatch(5).X | Post20thBatch(5).X >=
AreaT1X3(5);
       ix(58:76) = 1;
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
```

```
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);

    end
    if i == 6 % Cuts off backwards values for 6th termini
       clear ix
       ix =  AreaT20X3(6)  >= Post20thBatch(6).X | Post20thBatch(6).X >=
AreaT1X3(6);
       ix(58:93) = 1;
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);

    end
    if i == 7 % Cuts off backwards values for 6th termini
       clear ix
       ix =  AreaT20X3(7)  >= Post20thBatch(7).X | Post20thBatch(7).X >=
AreaT1X3(7);
       ix(42:76) = 1; % Change this every termini sample
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);

    end
    if i == 8 % Cuts off backwards values for 6th termini
       clear ix
```

```
       ix =  AreaT20X3(8)  >= Post20thBatch(8).X | Post20thBatch(8).X >=
AreaT1X3(8);
       ix(75:112) = 1; % Change this every termini sample
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);

    end
    if i == 9 % Cuts off backwards values for 6th termini
       clear ix
       ix =  AreaT20X3(9)  >= Post20thBatch(9).X | Post20thBatch(9).X >=
AreaT1X3(9);
       ix(49:73) = 1; % Change this every termini sample
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);

    end
    if i == 10 % Cuts off backwards values for 6th termini
       clear ix
       ix =  AreaT20X3(10)  >= Post20thBatch(10).X | Post20thBatch(10).X >=
AreaT1X3(10);
       ix(44:73) = 1; % Change this every termini sample
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
```

```
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);

    end
    if i == 11 % Cuts off backwards values for 6th termini
       clear ix
       ix =  AreaT20X3(11)  >= Post20thBatch(11).X | Post20thBatch(11).X >=
AreaT1X3(11);
       ix(48:77) = 1; % Change this every termini sample
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);
    end
    if i == 12 % Cuts off backwards values for 6th termini
       clear ix
       ix =  AreaT20X3(12)  >= Post20thBatch(12).X | Post20thBatch(12).X >=
AreaT1X3(12);
       ix(58:78) = 1; % Change this every termini sample
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);
    end
    if i == 13 % Cuts off backwards values for 6th termini
       clear ix
       ix =  AreaT20X3(13)  >= Post20thBatch(13).X | Post20thBatch(13).X >=
AreaT1X3(13);
       ix(42:78) = 1; % Change this every termini sample
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
```

```
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);
    end
    if i == 14 % Cuts off backwards values for 6th termini
       clear ix
       ix =  AreaT20X3(14)  >= Post20thBatch(14).X | Post20thBatch(14).X >=
AreaT1X3(14);
       ix(43:70) = 1; % Change this every termini sample
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);
    end
    if i == 15 % Cuts off backwards values for 6th termini
       clear ix
       ix =  AreaT20X3(15)  >= Post20thBatch(15).X | Post20thBatch(15).X >=
AreaT1X3(15);
       ix(39:57) = 1; % Change this every termini sample
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);
    end
    if i == 16 % Cuts off backwards values for 6th termini
       clear ix
       ix =  AreaT20X3(16)  >= Post20thBatch(16).X | Post20thBatch(16).X >=
AreaT1X3(16);
       ix(58:70) = 1; % Change this every termini sample
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
```

```
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('Latitude')
        ylabel('Longitude')
        Area3(i) = polyarea(AllX, AllY);
    end
    if i == 17 % Cuts off backwards values for 6th termini
       clear ix
       ix =  AreaT20X3(17)  >= Post20thBatch(17).X | Post20thBatch(17).X >=
AreaT1X3(17);
       ix(77:95) = 1; % Change this every termini sample
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('NZTM X (meters)')
        ylabel('NZTM Y (meters)')
        Area3(i) = polyarea(AllX, AllY);
    end
    if i == 18 % Cuts off backwards values for 6th termini
       clear ix
       ix =  AreaT20X3(18)  >= Post20thBatch(18).X | Post20thBatch(18).X >=
AreaT1X3(18);
       ix(58:67) = 1; % Change this every termini sample
       resultX = Post20thBatch(i).X;
        resultX(ix) = NaN;
        resultY = Post20thBatch(i).Y;
        resultY(ix) = NaN;
        resultX(find(isnan(resultX))) = []
        resultY(find(isnan(resultY))) = []
        AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i),
BoundaryX1, BoundaryX20]; % Here i create a X vector with points that can be joined
up to ascertain an area
        AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i),
BoundaryY1, BoundaryY20]; % Has The terminus, the vertices of the intersections
        plot(AllX, AllY)
        %plot(AllXmax, AllYmax)
        text(Lx4-50, Ly4+30, str2);
        grid off
        xlabel('NZTM X (meters)')
        ylabel('NZTM Y (meters)')
        Area3(i) = polyarea(AllX, AllY);
    end

    Area3
end
for i = 19:Size3(1)
    clear resultX
    clear resultY
```

```
    clear ix
    ix =  AreaT20X3(i)  >= Post20thBatch(i).X | Post20thBatch(i).X >= AreaT1X3(i);
    resultX = Post20thBatch(i).X;
    resultX(ix) = NaN;
    resultY = Post20thBatch(i).Y;
    resultY(ix) = NaN;
    % Remove all entries that are NaN - this was causing problems where the
    % lines did not join in polyarea.
    resultX(find(isnan(resultX))) = []
    resultY(find(isnan(resultY))) = []

    % Counter clockwise = + area , clockwise = - area , currently resultx =
    % Left to right - Vector creation here
    AllX = [ BoundaryX20, AreaT20X3(i), fliplr(resultX),  AreaT1X3(i), BoundaryX1,
BoundaryX20]; % Here i create a X vector with points that can be joined up to
ascertain an area
    AllY = [ BoundaryY20, AreaT20Y3(i), fliplr(resultY), AreaT1Y3(i), BoundaryY1,
BoundaryY20]; % Has The terminus, the vertices of the intersections
    hold on

    plot(AllX, AllY)
    text(Lx4-50, Ly4+30, str2);
    grid off
    title('Batch Three Area Measurements','Fontsize',20)
    xlabel('NZTM X (meters)')
    ylabel('NZTM Y (meters)')

    Area3(i) = polyarea(AllX, AllY);
    Area3

end
hold off
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% SURFACE AREA %%%%%%%%%%%%%%%
%%% BATCH FOUR %%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
for j = 1:Size4(1);
    if LastBatch(j).X(2)< LastBatch(j).X(end-1)
       LastBatch(j).X = fliplr(LastBatch(j).X); % Sorts it based on descending
order
       LastBatch(j).Y = fliplr(LastBatch(j).Y); % Switches y to Left to Right using
fliplr command
    else
       LastBatch(j).X = LastBatch(j).X;
       LastBatch(j).Y = LastBatch(j).Y;
    end
end
% Batch 4
for i = 1:Size4(1); % Number of termini in total
    clear resultX
    clear resultY
    ix =  AreaT20X4(i) >= LastBatch(i).X | LastBatch(i).X >= AreaT1X4(i); % This
makes a 1 for all those that satisfy this condition (FINAL CHECK)
    resultX = LastBatch(i).X;
    resultX(ix) = NaN;
    resultY = LastBatch(i).Y;
    resultY(ix) = NaN;
    % Remove all entries that are NaN - this was causing problems where the
    % lines did not join in polyarea.
    resultX(find(isnan(resultX))) = []
    resultY(find(isnan(resultY))) = []
    % Counter clockwise = + area , clockwise = - area , currently resultx =
    % Left to right - Vector creation here
    AllX = [ BoundaryX20, AreaT20X4(i), fliplr(resultX),  AreaT1X4(i), BoundaryX1,
BoundaryX20]; % Here i create a X vector with points that can be joined up to
ascertain an area
```

```
    AllY = [ BoundaryY20, AreaT20Y4(i), fliplr(resultY), AreaT1Y4(i), BoundaryY1,
BoundaryY20]; % Has The terminus, the vertices of the intersections
%     AllX = [ BoundaryX20, AreaT20X1(i), resultX, AreaT1X1(i), BoundaryX1,
BoundaryX20]; % Here i create a X vector with points that can be joined up to
ascertain an area
%     AllY = [ BoundaryY20, AreaT20Y1(i), resultY, AreaT1Y1(i), BoundaryY1,
BoundaryY20];
    hold on
    subplot(2,2,4)
    plot(AllX, AllY)
    %plot(AllXmax, AllYmax)
    text(Lx4-50, Ly4+30, str2);
    grid off
    title('Batch Four Area Measurements', 'Fontsize',20)
    xlabel('NZTM X (meters)')
    ylabel('NZTM Y (meters)')
    hold off
    Area4(i) = polyarea(AllX, AllY);
    Area4

end
%Area4 = Area4(3:end);
AreaAll = [Area1, Area2, Area3, Area4];
AreaAllInt = int64(AreaAll); % THIS IS BECAUSE MATLAB LIKES TO STORE INFORMATION AS
DOUBLES...convert to integer to see true area
%%
%%% CALCULATIONS %%%
%%% NET LOSS FROM 3 events %%%
%%% Large calving
Event1 = AreaAll(3) - AreaAll(4)
Event2 = AreaAll(139) - AreaAll(140)
Event3 = AreaAll(146) - AreaAll(147)

%%% BACKGROUND Calving
Background1 = ((AreaAll(1) - AreaAll(3)) + (AreaAll(4) - AreaAll(57)))*1e6 %0.0156
Background2 = ((AreaAll(57) - AreaAll(94)))*1e6 % 0.0417
Background3 = ((AreaAll(94) - AreaAll(139)) + (AreaAll(140) - AreaAll(146)) +
(AreaAll(147) - AreaAll(end-9)))*1e6


EventAll = Event1+ Event2+ Event3
EventAllkm = EventAll * 1e-6
EventPerc0 = EventAll/NetSALossM
EventPerc1 = Event1/NetSALossM
EventPerc2 = Event2/NetSALossM
EventPerc3 = Event3/NetSALossM

%%% Small calving
Event4 = (Area3(35) - Area3(53))*1e6 % 2015 Summer from Start of Decemeber 2014 to
end of February 2015
EventPerc4 = Event4/NetSALossM
%%% 0.0368 km squared
% -----------------------------
%%% ADVANCE EVENTS %%%
Advance1 = ((Area1(9) - Area1(10))/((Area1(1)-Area1(end))
Advance2 = (((Area2(1) - Area2(4))*(Datenm2(1)-Datenm3(4)))+(Area2(5)-
Area2(6))*(Datenm2(4)-Datenm3(5))+(Area2(7)-Area2(8))+(Area2(23) -
Area2(25))))/(mean(Area2)*4)
Advance3 = ((0.0939 - 0.0946) + (0.0933 - 0.0944) + (0.0923 - 0.0946) + (0.0932-
0.0956) + (0.0915-0.0929) + (0.0908-0.0927))/ (Area3(1) - Area3(end))
Trapz1 = trapz(Area1)
Trapz2 = trapz(Area2)
Trapz3 = trapz(Area3)

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% PLOT ALL AREAS IN SUBPLOT %%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
width = 2; % Width of bar plots
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Allocate Max area %%%
%%%%%%%%%%%%%%%%%%%%%%%%%
clear Areamax1
clear Areamax2
clear Areamax3
clear Areamax4
%MaxArea = Area4(end);
Areamax1(1:length(Area1)) = MaxArea;
Areamax2(1:length(Area2)) = MaxArea;
Areamax3(1:length(Area3)) = MaxArea;
Areamax4(1:length(Area4)) = MaxArea;

% Transpose all Areas (Max subtracting every entry)
Area1 = Areamax1 - Area1;
Area2 = Areamax2 - Area2;
Area3 = Areamax3 - Area3;
Area4 = Areamax4 - Area4; % Run once

%%% CONVERT AREA to Km Squared
Area1 = Area1 .* 1e-6
Area2 = Area2 .* 1e-6
Area3 = Area3 .* 1e-6
Area4 = Area4 .* 1e-6

%%%
figure (100)
fig = figure;
left_color = [.5 .5 0]; %%% CHANGE GRAPH COLOR HERE
right_color = [0 .5 .5]; %%% CHANGE GRAPH COLOR HERE
set(fig,'defaultAxesColorOrder',[left_color; right_color]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(2,2,1)
hold on
%stairs(Datenm1, Area1, 0.5,'r')
yyaxis right % define right axis plot
plot(Datenm1, IntFinalAv1) %% GRAY COLOR
%set(p,'FaceAlpha',0.125, 'EdgeColor', 'none') % Plot retreat rate in background %%
Alphas command
ylabel ('Meters of Retreat (m)','Fontsize', 12)
%%%
yyaxis left
plot(Datenm1, Area1,':^')
xlabel('2013','Fontsize', 12)
ylabel('Surface Area (km squared)','Fontsize', 12)

title('Surface Area flux for Batch One','Fontsize', 20)
legend( 'Surface Area Flux (Batch One)','Averaged terminus positions (m)')
datetick('x')
hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(2,2,2)
hold on
yyaxis right % define right axis plot
plot(Datenm2, IntFinalAv2)
ylabel ('Meters of Retreat (m)','Fontsize', 12)
yyaxis left
plot(Datenm2, Area2,':^')
datetick('x')
xlabel('2013','Fontsize', 12)
ylabel('Surface Area (km squared)','Fontsize', 12)
ylim([0 0.25])
title('Surface Area flux for Batch Two','Fontsize', 20)
legend( 'Surface Area Flux (Batch Two)','Averaged terminus positions (m)')
axis tight
hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
subplot(2,2,3)
```

```
hold on
%%% Fix duplicates issue with Bar (future idea is using unique function) -
%%% FOUND ISSUE In ENTRY 43 and 44 of Datenm3. The date is therefore
%%% overlapping. Therefore Deleting entry 44 in Datenm3 AND Area4
%% --------------------------------
%Datenm3 = Datenm3([1:43, 45:end]);
%Area3 = Area3([1:43, 45:end]);
%IntFinalAv3 = IntFinalAv3([1:43, 45:end]);
%% --------------------------------
yyaxis right
plot(Datenm3, IntFinalAv3)
ylabel ('Meters of Retreat (m)','Fontsize', 12)
yyaxis left
plot(Datenm3, Area3,':^')
datetick('x')
xlabel('2013-2015','Fontsize', 12)
ylabel('Surface Area (km squared)','Fontsize', 12)
ylim([0 0.25])
title('Surface Area flux for Batch Three','Fontsize', 20)
legend( 'Surface Area Flux (Batch Three)','Averaged terminus positions (m)')
axis tight
hold off

subplot(2,2,4)
hold on
yyaxis right
plot(Datenm4, IntFinalAv4)
ylabel ('Meters of Retreat (m)','Fontsize', 12)
yyaxis left
plot(Datenm4, Area4,':^')
datetick('x')
xlabel('2015-2016', 'Fontsize', 12)
ylabel('Surface Area (km squared)','Fontsize', 12)
ylim([0 0.25])
title('Surface Area flux for Batch Four','Fontsize', 20)
legend( 'Surface Area Flux (Batch Four)','Averaged terminus positions (m)')
axis tight
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLOT ALL AREA in 1 BAR GRAPH
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

AreaAll = [Area1, Area2, Area3, Area4]; % All Areas
DateAll1 = [Datenm1, Datenm2, Datenm3, Datenm4]; % All Dates
% IntAllAverage = IntAllAverage([1:88, 90:end]); % Remove the 1 bad Datenm3 entry
at 89

fig = figure;
left_color = [.5 .5 0]; %%% CHANGE GRAPH COLOR HERE
right_color = [0 .5 .5]; %%% CHANGE GRAPH COLOR HERE
set(fig,'defaultAxesColorOrder',[left_color; right_color]);
hold on
yyaxis right
plot(DateAll1, IntAllAverage)
ylabel ('Meters of Retreat (m)','Fontsize', 12)
yyaxis left
plot(DateAll1, AreaAll,':^')
datetick('x','dd-mmm-yyyy', 'keepticks')
xlabel('2013-2016', 'Fontsize', 12)
ylabel('Surface Area (km squared)','Fontsize', 12)
ylim([-0.025 0.25])
title('Minimum Surface Area flux for all Batches','Fontsize', 16)
legend( 'Surface Area Flux (All)','Averaged terminus positions (m)')

hold off

%%%%%%%%%%%%%%%%%%
%%% CALCULATIONS
```

```
%%% 2013-2014
SA2013 = Area1(1) - Area3(12) % THIS MEANS THE END SAMPLE OF 2013 is SAMPLE 45+12 =
57
%%% 2014-2015
SA2014 = Area3(12) - Area3(49) % END SAMPLE OF 2014 IS 45 + 49 = 94
%%%% 2015-2016
Sa2015 = Area3(49) - Area4(end) % END SAMPLE OF 2015 is end - 9
%%%% 2015 - 2016 EXACT ONE YEAR
Sa2015b = Area3(49) - Area4(end-9)





%%%%%%%%%%%%%%%%%%%%%%%
% PLOT AREA RESULTS (Lake Area vs Tasman Area)
%%%%%%%%%%%%%%%%%%%%%%%

figure(23)
hold on
plot(DateAll1,AreaAllInt, 'Color', MATC4)
datetick ('x',  'dd-mmm-yyyy', 'keepticks');
title ('Tasman Lake Surface Area Flux (2013-2016)')
xlabel('Month and Year')
ylabel('Meters squared (m^2)')
scatter(AllX, AllY)
plot(DateAll1, AreaAll, 'Color', MATC2)
hold off

figure(24)
% Invert area loss into a subtraction of the previous amount from TOTAL AREA
TotalAreaX = [LastBatch(end).X(1:end-1), BoundaryX1,  BoundaryX20];
TotalAreaY = [LastBatch(end).Y(1:end-1), BoundaryY1, BoundaryY20];
TotalArea = polyarea(TotalAreaX, TotalAreaY);

FinalAreaAll = TotalArea - AreaAll;
plot(DateAll1,FinalAreaAll, 'r')
datetick ('x',  'dd-mmm-yyyy', 'keepticks');
title ('Tasman Surface Area Flux Transposed (2013-2016)')
xlabel('Month and Year')
ylabel('Meters squared (m^2)')





%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%
% (APPENDIX of Code) Use intersection to find intersection of the terminus
positions
% crossed with the transects created

% Calculating Area
% Quadvec (aka two dimensional integration over a general domain) doesnt work
because it requires a FUNCTION, which means the
% points have already been mapped out as a mathemtical FUNCTION , Which is
% not possible. Inpolygon works only if you can define the 'shape' of the
% polygon you wish to capture points within, therefore once again requiring
% the capture of points in such a regard.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ALTERNATIVE THINKING FROM INTERSECTIONS
% Find the minimum distance between the clicked location and the data points
%[a,b]=min((xc-x).^2+(yc-y).^2);
% Output the results -- using b(1) in case we get exactly between two
```

```
% points. [VERY unlikely, but theoretically possible.]
%disp(['The closest point to the one you clicked was x=' num2str(xc(b(1)))',
%y =' num2str(yc(b(1))) '.'])

%str=num2str(Po);
%text(Date_First(1,:),Date_Second(1,:), Date_Third(1,:), str 'FontSize', 8); %
Display the times .. still not working

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (6) Loop to calculate the average width
% First Batch - Save Each
% [X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18,
X19, X20] = FirstBatch.X; % 0 - 20
% [X21, X22, X23, X24, X25, X26, X27, X28, X29, X30, X31, X32, X33, X34, X35, X36,
X37, X38, X39, X40, X41, X42, X43, X44, X45 ] = Pre20thBatch.X; % 21-45
% [X46, X47, X48, X49, X50, X51, X52, X53, X54, X55, X56, X57, X58, X59, X60, X61,
X62, X63, X64, X65, X66, X67, X68, X69, X70, X71, X72, X73, X74, X75, X76, X77,
X78, X79, X80, X81, X82, X83, X84, X85, X86, X87, X88, X89, X90, X91, X92, X93,
X94, X95, X96, X97, X98, X99, X100, X101, X102, X103, X104, X105, X106, X107, X108,
X109, X110, X111, X112, X113, X114, X115, X116, X117, X118, X119, X120, X121, X122,
X123, X124, X125, X126, X127, X128, X129, X130] = Post20thBatch.X; %46-130
% [X131, X132, X133, X134, X135, X136, X137, X138, X139, X140, X141, X142, X143,
X144, X145, X146, X147, X148, X149, X150, X151, X152, X153, X154, X155, X156] =
LastBatch.X %131 - 156
%
% [Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y8, Y9, Y10, Y11, Y12, Y13, Y14, Y15, Y16, Y17, Y18,
Y19, Y20] = FirstBatch.Y; % 0 - 20
% [Y21, Y22, Y23, Y24, X25, X26, X27, X28, X29, X30, X31, X32, X33, X34, X35, X36,
X37, X38, X39, X40, X41, X42, X43, X44, X45 ] = Pre20thBatch.X; % 21-45
% [X46, X47, X48, X49, X50, X51, X52, X53, X54, X55, X56, X57, X58, X59, X60, X61,
X62, X63, X64, X65, X66, X67, X68, X69, X70, X71, X72, X73, X74, X75, X76, X77,
X78, X79, X80, X81, X82, X83, X84, X85, X86, X87, X88, X89, X90, X91, X92, X93,
X94, X95, X96, X97, X98, X99, X100, X101, X102, X103, X104, X105, X106, X107, X108,
X109, X110, X111, X112, X113, X114, X115, X116, X117, X118, X119, X120, X121, X122,
X123, X124, X125, X126, X127, X128, X129, X130] = Post20thBatch.X; %46-130
% [X131, X132, X133, X134, X135, X136, X137, X138, X139, X140, X141, X142, X143,
X144, X145, X146, X147, X148, X149, X150, X151, X152, X153, X154, X155, X156] =
LastBatch.X %131 - 156
%
% for i = 1:20
%
% end
```

StandardErrorUr.m

This code is used for post-processing my results of the Terminus Vectorisation stage. I calculated the standard deviation and standard errors for the retreat rate approximations (U_r) presented in the results.

```
% Test for Std
% A = [1,2,3,4,5,6,7,8,9,10]
% MeanTest = mean(A)
% STDTest = std(A)
% TestMeansFromSamples = [10,11,12,13,14,15,16,17];
% n = length(TestMeansFromSamples)
% STDERRORTest = sqrt(1/n) * ((10-MeanTest)^2 + (11-Meantest)^2 + (12-
MeanTest)^2

% Where Samples are each TRANSECT virtually
% First calculate all Values from the Transects to attain Standard Error
for
% retreat rates
% This is the distance from IntFinalA1(1) - IntFinalA1(end)
% Then store this into a vector called AllRates
```

```
%%% Batch One %%%%
r =  IntFinal1(1)-IntFinal1(end)
r2 =  IntFinal2(1)-IntFinal2(end)
r3 =  IntFinal3(1)-IntFinal3(end)
r4 =  IntFinal4(1)-IntFinal4(end)
r5 =  IntFinal5(1)-IntFinal5(end)
r6 =  IntFinal6(1)-IntFinal6(end)
r7 =  IntFinal7(1)-IntFinal7(end)
r8 =  IntFinal8(1)-IntFinal8(end)
r9 =  IntFinal9(1)-IntFinal9(end)
r10 =  IntFinal10(1)-IntFinal10(end)
r11 =  IntFinal11(1)-IntFinal11(end)
r12 =  IntFinal12(1)-IntFinal12(end)
r13 =  IntFinal13(1)-IntFinal13(end)
r14 =  IntFinal14(1)-IntFinal14(end)
r15 =  IntFinal15(1)-IntFinal15(end)
r16 =  IntFinal16(1)-IntFinal16(end)
r17 =  IntFinal17(1)-IntFinal17(end)
r18 =  IntFinal18(1)-IntFinal18(end)
r19 =  IntFinal19(1)-IntFinal19(end)
r20 =  IntFinal20(1)-IntFinal20(end)
AllMeans1 =
[r,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r15,r16,r17,r18,r19,r20];
StdAllMeans1 = std(AllMeans1)
StdErr1 = sqrt(1/20) * StdAllMeans1

%%% BATCH TWO %%%
r =  IntFinalB1(1)-IntFinalB1(end)
r2 =  IntFinalB2(1)-IntFinalB2(end)
r3 =  IntFinalB3(1)-IntFinalB3(end)
r4 =  IntFinalB4(1)-IntFinalB4(end)
r5 =  IntFinalB5(1)-IntFinalB5(end)
r6 =  IntFinalB6(1)-IntFinalB6(end)
r7 =  IntFinalB7(1)-IntFinalB7(end)
r8 =  IntFinalB8(1)-IntFinalB8(end)
r9 =  IntFinalB9(1)-IntFinalB9(end)
r10 =  IntFinalB10(1)-IntFinalB10(end)
r11 =  IntFinalB11(1)-IntFinalB11(end)
r12 =  IntFinalB12(1)-IntFinalB12(end)
r13 =  IntFinalB13(1)-IntFinalB13(end)
r14 =  IntFinalB14(1)-IntFinalB14(end)
r15 =  IntFinalB15(1)-IntFinalB15(end)
r16 =  IntFinalB16(1)-IntFinalB16(end)
r17 =  IntFinalB17(1)-IntFinalB17(end)
r18 =  IntFinalB18(1)-IntFinalB18(end)
r19 =  IntFinalB19(1)-IntFinalB19(end)
r20 =  IntFinalB20(1)-IntFinalB20(end)
AllMeans2 =
[r,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r15,r16,r17,r18,r19,r20];
StdAllMeans2 = std(AllMeans2)
StdErr2 = sqrt(1/20) * StdAllMeans2



%%% BATCH THREE %%%
r =  IntFinalC1(1)-IntFinalC1(end)
r2 =  IntFinalC2(1)-IntFinalC2(end)
r3 =  IntFinalC3(1)-IntFinalC3(end)
r4 =  IntFinalC4(1)-IntFinalC4(end)
r5 =  IntFinalC5(1)-IntFinalC5(end)
r6 =  IntFinalC6(1)-IntFinalC6(end)
```

```
r7 =  IntFinalC7(1)-IntFinalC7(end)
r8 =  IntFinalC8(1)-IntFinalC8(end)
r9 =  IntFinalC9(1)-IntFinalC9(end)
r10 =  IntFinalC10(1)-IntFinalC10(end)
r11 =  IntFinalC11(1)-IntFinalC11(end)
r12 =  IntFinalC12(1)-IntFinalC12(end)
r13 =  IntFinalC13(1)-IntFinalC13(end)
r14 =  IntFinalC14(1)-IntFinalC14(end)
r15 =  IntFinalC15(1)-IntFinalC15(end)
r16 =  IntFinalC16(1)-IntFinalC16(end)
r17 =  IntFinalC17(1)-IntFinalC17(end)
r18 =  IntFinalC18(1)-IntFinalC18(end)
r19 =  IntFinalC19(1)-IntFinalC19(end)
r20 =  IntFinalC20(1)-IntFinalC20(end)
AllMeans3 =
[r,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r15,r16,r17,r18,r19,r20];
StdAllMeans3 = std(AllMeans3)
StdErr3 = sqrt(1/20) * StdAllMeans3

%%% BATCH FOUR FINAL BATCH %%%%%
r =  IntFinalD1(1)-IntFinalD1(end)
r2 =  IntFinalD2(1)-IntFinalD2(end)
r3 =  IntFinalD3(1)-IntFinalD3(end)
r4 =  IntFinalD4(1)-IntFinalD4(end)
r5 =  IntFinalD5(1)-IntFinalD5(end)
r6 =  IntFinalD6(1)-IntFinalD6(end)
r7 =  IntFinalD7(1)-IntFinalD7(end)
r8 =  IntFinalD8(1)-IntFinalD8(end)
r9 =  IntFinalD9(1)-IntFinalD9(end)
r10 =  IntFinalD10(1)-IntFinalD10(end)
r11 =  IntFinalD11(1)-IntFinalD11(end)
r12 =  IntFinalD12(1)-IntFinalD12(end)
r13 =  IntFinalD13(1)-IntFinalD13(end)
r14 =  IntFinalD14(1)-IntFinalD14(end)
r15 =  IntFinalD15(1)-IntFinalD15(end)
r16 =  IntFinalD16(1)-IntFinalD16(end)
r17 =  IntFinalD17(1)-IntFinalD17(end)
r18 =  IntFinalD18(1)-IntFinalD18(end)
r19 =  IntFinalD19(1)-IntFinalD19(end)
r20 =  IntFinalD20(1)-IntFinalD20(end)
AllMeans4 =
[r,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r15,r16,r17,r18,r19,r20];
StdAllMeans4 = std(AllMeans4)
StdErr4 = sqrt(1/20) * StdAllMeans4

%%% CALCULATE STANDARD DEVIATIONS FOR ANNUAL RATES %%%
%%% 2013-2014

YearStd1 = ( (70.6*20+4.04*20+23.41*20) / (20+20+20) )*0.5
YearError1 = YearStd1 * sqrt(1/20)

%%% 2014-2015

YearStd2 = StdAllMeans3
YearError2 = YearStd2 * sqrt(1/20)


%%% 2015-2016

YearStd3 = ( (23.41*20+66.74*20) / (20+20) )*0.5
YearError3 = YearStd3 * sqrt(1/20)
```

```
%%% ALL
YearStdAll =  ( (23.41*20+66.74*2 +70.6*20+4.04*20) / (80) )*0.5
YearErrorAll = YearStdAll * sqrt(1/20)


% Example for STD
%{ [SD12*n1+SD22*n2+SD32*n3] / [n1+n2+n3] }0.5.
%{ [70.6*20+4.04*20+23.41*20] / [20+20+20] }0.5
```

DirectionOfFlowAttain.m

This code was used to attain a direction of flow for the Tasman Glacier – used in the calculation of velocity from gridded sample points on the glacier. The GPS data was provided by Dr. Huw Horgan from one of this GPS setups on the Tasman Glacier. The data was for 2013-2015 only.

```
%%%% ATTAINING DIRECTION OF FLOW %%%%
%%% Source Huw's GPS data %%%%
%%%  dat1 = [t1, v1, vmin1, vmax1, vzi, vzmini, vzmaxi, vx1, vy1, dir1];
%%% Samples already taking a 24 hour median and captured and done this at
%%% every hourly timestep rather than at every 30 seconds

load C:\Users\Edmond\Desktop\THE THESIS\[7] Measuring velocity\arc1_all.txt

%%%% DIVIDE THE GPS TIMES %%%%
GPSDate = GPSDate/86400; % Because it is in seconds

%%% TEST PLOT %%%
plot (GPSDate, GPSdir1)
datetick('x', 'dd-mmm-yyyy', 'keepticks', 'keeplimits')

%%% FIND INTERSECTION BETWEEN DateAll1 AND GPS Date to find the averaged
%%% vectors for Vz, Vx and Vy

%%% CROP THE GPS DATE TO ONLY the first XXXXXX digits?
GPSDate = datestr(cellfun(@(x) x(1:10),GPSDate(cellfun('length',GPSDate) >
7),'un',0))
GPSDateVec = datevec(GPSDate) % Converts to a vector format for the dates
GPSDateNew = datenum(GPSDateVec(:,1:3)) % Crops only the first three
columns

% Problem is that intersect wont work unless it is exactly the same date,
% even at the seconds level
%[GPSInt, GPSia, GPSib]  = intersect(DateAll1, GPSDateNew) % Calculate
intersection
%GPSdateschosen = datestr(DateAll1(GPSia)) % Present all the dates that
have been chosen

%%%%%%%% GPSvx1 %%%%%%%%%%%%%%%%%%%%%%
GPSvxFull = GPSvx1(GPSib)

[U,IA,IC] = unique(GPSDateNew)
win = 5 % 5 days
%%%%% RUNNING MEAN of 5 Day Average %%%%%%%%%%
for n=1:length(Datenm1)
    dateON = 0.5 + Datenm1(n) - 0.5*(5) % Lower time bracket threshold
    dateOFF = 0.5 + Datenm1(n) + 0.5*(5) % Time bracket threshold
ix = find( GPSDateNew >= dateON &  GPSDateNew <= dateOFF)
vxNewAv(n)= nanmean(GPSvx1(ix))
vyNewAv(n)= nanmean(GPSvy1(ix))
vzNewAv(n)= nanmean(GPSvzi(ix))
```

```
end
% vxNewAv(end-1) = NaN
% vyNewAv(end-1) = NaN
% vzNewAv(end-1) = NaN
% vxNewAv(end) = NaN
% vyNewAv(end) = NaN
% vzNewAv(end) = NaN

for n=1:length(Datenm2)
    dateON = 0.5 + Datenm2(n) - 0.5*(5) % Lower time bracket threshold
    dateOFF = 0.5 + Datenm2(n) + 0.5*(5) % Time bracket threshold
ix = find( GPSDateNew >= dateON &  GPSDateNew <= dateOFF)
vxNewAv2(n)= nanmean(GPSvx1(ix))
vyNewAv2(n)= nanmean(GPSvy1(ix))
vzNewAv2(n)= nanmean(GPSvzi(ix))
end
figure,
hold on,
plot(Datenm1, vxNewAv,'b'), plot(Datenm2,vxNewAv2,'b')
plot(Datenm1, vyNewAv,'g'), plot(Datenm2,vyNewAv2,'g')
plot(Datenm1, vzNewAv,'r'), plot(Datenm2,vzNewAv2,'r')
datetick('x')
hold off

vxNewMean2 = nanmean(vxNewAv) %-6.592e-7
vyNewMean2 = nanmean(vyNewAv) %-1.449e-6
vzNewMean2 = nanmean(vzNewAv) %-1.4544e-7



%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%% INSTANTANEOUS GPS AT EACH TIMESTEP %%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% OLD METHODOLOGY WITH JUST INTERSECTIONS %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Convert all NaN to 0 in every GPS entry
notNaN = ~isnan(GPSvx1);
GPSvx1(~notNaN) = NaN;
notNaN = ~isnan(GPSvy1);
GPSvy1(~notNaN) = NaN;
notNaN = ~isnan(GPSvzi);
GPSvzi(~notNaN) = NaN;

vxNew2 = nanmean(GPSvx1(ix))
vyNew2 = nanmean(GPSvy1(ix))
vzNew2 = nanmean(GPSvzi(ix))
end
%%%% REMOVE ALL ANOMALIES %%%%
for i = 1:length(GPSvx1) %%% REMOVES THE 0.0018 entries
    if GPSvx1(i) >= 0.001
        GPSvx1(i) = 0
    end
end
plot(GPSDateNew, GPSvx1) % TO SEE WHAT IS GOING ON
datetick('x')
%%%%%%% GPSvx1 %%%%%%%%%%%%%%%%%%%%%%
vxNew = interp1(GPSDateNew(IA), GPSvx1(IA), DateAll1) % Interpolate to find
the resampled amount at the dates
vxNewMean = nanmean(vxNew); % 5.8718e7
vxNew2 = nanmean(GPSvx1) %Average across ALL entries % -3.366e-7
%%%%%%% GPSvy1 %%%%%%%%%%%%%%%%%%%%%%
```

```
vyNew = interp1(GPSDateNew(IA), GPSvy1(IA), DateAll1)
vyNewMean = nanmean(vyNew); % -8.7926e-7
vyNew2 = nanmean(GPSvy1) %Average across ALL entries % 1.6687e-6
%%%%%%% GPSvz1 %%%%%%%%%%%%%%%%%%%%%%
vzNew = interp1(GPSDateNew(IA), GPSvzi(IA), DateAll1)
vzNewMean = nanmean(vzNew); % -1.3493e-7
vzNew2 = nanmean(GPSvzi) %Average across ALL entries % -1.1852e-7


vxNew2
vyNew2
vzNew2

%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% PLOT DIRECTION RESULTS FROM GPS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(105)
%vxNew(20) = NaN;
hold on
axis tight
plot(DateAll1, vxNew,'b-')
%ax.XTick = [DateAll1(1) DateAll1(38) DateAll1(76) DateAll1(114)
DateAll1(end)];
datetick('x', 'dd-mmm-yyyy' )

vxMean = nanmean(vxNew);
plot(DateAll1, vxNewMean*ones(size(DateAll1)),'b:') % Plots a straight line
horizontal. It needs to have "ones"

vyNew(20) = NaN
plot(DateAll1, vyNew,'g-')
datetick('x', 'dd-mmm-yyyy')
%ax.XTick = [DateAll1(1) DateAll1(38) DateAll1(76) DateAll1(114)
DateAll1(end)];
vyMean = nanmean(vyNew);

% Convert NaN to 0
vyMean = nanmean(vyNew);
plot(DateAll1, vyNewMean*ones(size(DateAll1)),'g:')


plot(DateAll1, vzNew,'r-')
datetick('x', 'dd-mmm-yyyy' )
%ax.XTick = [DateAll1(1) DateAll1(38) DateAll1(76) DateAll1(114)
DateAll1(end)];

vzMean = nanmean(vzNew);
plot(DateAll1, vzNewMean*ones(size(DateAll1)),'r:')

ylabel ('X, y and z direction vectors (m)')
xlabel ('Months and Years')
title ('Direction vectors from Tasman GPS Station')
legend('vX', 'vX mean', 'vY', 'vY mean', 'vZ', 'vZ mean')

hold off
%%
```

VelocityConversionFinal.m

This is the final piece of code I used to convert the Digital Image Correlation output data (from Jones (2016)'s algorithm) to real-world positions using the Geometric Scale Factor (geom_scale) and the Pixel Scale Factors (px_size). Positions from the 119 world-grid generated sample points are then plotted againsts the DIC grid points (8816 for Batch One, 9282 for Batches 2 and 3 and 9401 for Batch 4 and 5) and the respective position data is translated and saved and concatenated. The magnitude of the position changes is calculated simply by the sqrt(x^2 + y^2). Moreover, the position matrices had to be concatenated at some point and to ensure compatibility, a temporary matrix of size 1 x length(cell) of 0s was created at each NaN entry.

The final output shows a timeseries of position relative to the base image (first image of the batch or first image of the plot). A control was set up by allocating all sample points that no lie on the Tasman Glacier (because the grid points don't move, but the glacier retreats over time). The control (cyan plots) are indicative of the overall camera movement that is affecting the positions over time. The Errors for this stage are an addition of the GSF scale errors during reprojection, the Digital Image Correlation Coefficient (scale 0 – 1, 1 for very good correlation) and the error from camera movement which was < 1 m for the entire series (max. at ~ 0.96 m in Batch Five). In total, five grouped batches were used in this phase, although a total of 16 DIC runs were made in aggregate) to minimise correlation anomalies and reduce errors.

```
%%%%%%%%%%%%%%%% VELOCITY from PIXEL WORLD to %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% VELOCITY IN REAL WORLD %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% All Batch Correlations are stored in valid_data.mats in each respective
%%% Batch Folder location
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%  BATCH ONE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% BATCH ONE - 40 m grid spacing and 51 subset spacing

load('valid_data.mat')
load('grid_data.mat')s
validx % Dimensions are pixel positions/movement (number of pixel --- 8816
for instance) v time (20)
validy % Dimensions are pixel positions/movement (number of pixel --- 8816
for instance) v time (20)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Access the respective image pixel location for each Velocity Sample
%%%% PLOT SCRIPT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Point %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Batch One (import data first) - IMPORT IT FROM the GSF FILE FOR EACH
%%% BATCH ------------------------------ IN THE FOLLOWING FORMATS
%%% VelNum, VelGSF, VelPXF, velxi, Velyi
% ------------------------------------------------------------
```

```
%%% GRID DIMENSIONS 76 y x 116 x
BatchOneSample = imread('20130205154143.tif');
BatchOneSampleGray = rgb2gray(BatchOneSample(:,:,1:3));
figure(1),
imshow(BatchOneSampleGray)
hold on
%load ('grid_data.mat') % In Jones(2015)_DIC/DIC Files/1/A - 51 Subset 50
Grid Serial Preceding
scatter(gridx, gridy, 'yx') % Plot all grid points for Correlation
%%% HOW DID I GET THE VELXi and VELYi SAMPLE POINT LOCATIONS? - in GIS?
%%% REDO THIS FOR BATCH TWO and THREE and FOUR
scatter(Velxi(1:39), Velyi(1:39), 'gx')% Plot velocity sample points
scatter(Velxi(40:79), Velyi(40:79), 'bx')
scatter(Velxi(80:120), Velyi(80:120), 'rx')

text(gridx(1), gridy(1), '0', 'Fontsize', 8)
text(gridx(end)/2, gridy(1), '58','Fontsize', 8,'Fontsize', 8)
text(gridx(end)/4, gridy(1), '29','Fontsize', 8)
text(gridx(end)*3/4, gridy(1), '87','Fontsize', 8)
text(gridx(end), gridy(1), '116','Fontsize', 8)
text(gridx(1), gridy(end)/4, '19','Fontsize', 8)
text(gridx(1), gridy(end)/2, '38','Fontsize', 8)
text(gridx(1), gridy(end)*3/4, '57','Fontsize', 8)
text(gridx(1), gridy(end), '76','Fontsize', 8)
text(gridx(1), gridy(end)*5/8, '47', 'Fontsize', 8)
text(gridx(end)-888, gridy(end)+20, 'Total number = 8816')
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Access the respective image pixel location for each Velocity Sample
%%%% PLOT SCRIPT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Batch2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BatchTwoSample = imread('20130723130033.jpg');
BatchTwoSampleGray = rgb2gray(BatchTwoSample(:,:,1:3));
figure(1),
imshow(BatchTwoSampleGray)
hold on

% load ('grid_data.mat') % In Jones(2015)_DIC/DIC Files/2/A - 51 Subset 50
Grid Serial Preceding
% gridx2 = gridx
% gridy2 = gridy
scatter(gridx2, gridy2, 'yx') % Plot all grid points for Correlation
%%% HOW DID I GET THE VELXi and VELYi SAMPLE POINT LOCATIONS? - in GIS?
%%% REDO THIS FOR BATCH TWO and THREE and FOUR
scatter(Velxi2(1:39), Velyi2(1:39), 'gx')% Plot velocity sample points
scatter(Velxi2(40:79), Velyi2(40:79), 'bx')
scatter(Velxi2(80:120), Velyi2(80:120), 'rx')
%%% GRIDS FROM THE DIC 119 x * 78 y = 9282
text(gridx2(1), gridy2(1), '0', 'Fontsize', 8)
text(gridx2(end)/2, gridy2(1), '60','Fontsize', 8,'Fontsize', 8)
text(gridx2(end)/4, gridy2(1), '30','Fontsize', 8)
text(gridx2(end)*3/4, gridy2(1), '90','Fontsize', 8)
text(gridx2(end), gridy2(1), '116','Fontsize', 8)
text(gridx2(1), gridy2(end)/4, '19','Fontsize', 8)
text(gridx2(1), gridy2(end)/2, '39','Fontsize', 8)
text(gridx2(1), gridy2(end)*3/4, '58','Fontsize', 8)
text(gridx2(1), gridy2(end), '78','Fontsize', 8)
text(gridx2(1), gridy2(end)*5/8, '49', 'Fontsize', 8)
text(gridx2(end)-888, gridy2(end)+20, 'Total number = 9282')
hold off
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Access the respective image pixel location for each Velocity Sample
%%%% PLOT SCRIPT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Batch3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BatchThreeSample = imread('20131209160032.jpg');
figure(1),
BatchThreeSampleGray = BatchThreeSample;
imshow(BatchThreeSampleGray) % Already in GrayScale
hold on

% load ('grid_data.mat') % In Jones(2015)_DIC/DIC Files/2/A - 51 Subset 50
Grid Serial Preceding
% gridx3 = gridx
% gridy3 = gridy
scatter(gridx3, gridy3, 'yx') % Plot all grid points for Correlation %%%%
% 9282 GRID POINTS IN TOTAL - IMPORT POSITIONS FROM GSF3 in the GSF Output
scatter(Velxi3(1:39), Velyi3(1:39), 'gx')% Plot velocity sample points
scatter(Velxi3(40:79), Velyi3(40:79), 'bx')
scatter(Velxi3(80:120), Velyi3(80:120), 'rx')
%%% GRIDS FROM THE DIC
text(gridx3(1), gridy3(1), '0', 'Fontsize', 8)
text(gridx3(end)/2, gridy3(1), '60','Fontsize', 8,'Fontsize', 8)
text(gridx3(end)/4, gridy3(1), '30','Fontsize', 8)
text(gridx3(end)*3/4, gridy3(1), '90','Fontsize', 8)
text(gridx3(end), gridy3(1), '116','Fontsize', 8)
text(gridx3(1), gridy3(end)/4, '19','Fontsize', 8)
text(gridx3(1), gridy3(end)/2, '39','Fontsize', 8)
text(gridx3(1), gridy3(end)*3/4, '58','Fontsize', 8)
text(gridx3(1), gridy3(end), '78','Fontsize', 8)
text(gridx3(1), gridy3(end)*5/8, '49', 'Fontsize', 8)
text(gridx3(end)-888, gridy3(end)+20, 'Total number = 9282')
hold off


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Access the respective image pixel location for each Velocity Sample
%%%% PLOT SCRIPT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Batch4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BatchFourSample = imread('20140218150031.jpg');
figure(1),
BatchFourSampleGray = BatchFourSample;
imshow(BatchFourSampleGray) % Already in Grayscale
hold on

% load ('grid_data.mat') % In Jones(2015)_DIC/DIC Files/2/A - 51 Subset 50
Grid Serial Preceding
% gridx4 = gridx
% gridy4 = gridy
scatter(gridx4, gridy4, 'yx') % Plot all grid points for Correlation %%%%
% 9282 GRID POINTS IN TOTAL - IMPORT POSITIONS FROM GSF4 in the GSF Output
scatter(Velxi4(1:39), Velyi4(1:39), 'gx')% Plot velocity sample points
scatter(Velxi4(40:79), Velyi4(40:79), 'bx')
scatter(Velxi4(80:120), Velyi4(80:120), 'rx')
%%% GRIDS FROM THE DIC % 119 x * 79 y = 9401
text(gridx4(1), gridy4(1), '0', 'Fontsize', 8)
text(gridx4(end)/2, gridy4(1), '60','Fontsize', 8,'Fontsize', 8)
text(gridx4(end)/4, gridy4(1), '30','Fontsize', 8)
text(gridx4(end)*3/4, gridy4(1), '90','Fontsize', 8)
text(gridx4(end), gridy4(1), '119','Fontsize', 8)
```

```
text(gridx4(1), gridy4(end)/4, '19','Fontsize', 8)
text(gridx4(1), gridy4(end)/2, '39','Fontsize', 8)
text(gridx4(1), gridy4(end)*3/4, '58','Fontsize', 8)
text(gridx4(1), gridy4(end), '78','Fontsize', 8)
text(gridx4(1), gridy4(end)*5/8, '49', 'Fontsize', 8)
text(gridx4(end)-888, gridy4(end)+20, 'Total number = 9401')
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Access the respective image pixel location for each Velocity Sample
%%%% PLOT SCRIPT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Batch5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BatchFiveSample = imread('20150827120026.jpg');
figure(1),
BatchFiveSampleGray = BatchFiveSample;
imshow(BatchFiveSampleGray) % Already in Grayscale
hold on

% load ('grid_data.mat') % In Jones(2015)_DIC/DIC Files/2/A - 51 Subset 50
Grid Serial Preceding
% gridx5 = gridx
% gridy5 = gridy
scatter(gridx5, gridy5, 'yx') % Plot all grid points for Correlation %%%%
% 9282 GRID POINTS IN TOTAL - IMPORT POSITIONS FROM GSF4 in the GSF Output

scatter(Velxi4(1:39), Velyi4(1:39), 'gx')% Plot velocity sample points
scatter(Velxi4(40:79), Velyi4(40:79), 'bx')
scatter(Velxi4(80:120), Velyi4(80:120), 'rx')

%%% GRIDS FROM THE DIC % 119 x * 79 y = 9401
text(gridx5(1), gridy5(1), '0', 'Fontsize', 8)
text(gridx5(end)/2, gridy5(1), '60','Fontsize', 8,'Fontsize', 8)
text(gridx5(end)/4, gridy5(1), '30','Fontsize', 8)
text(gridx5(end)*3/4, gridy5(1), '90','Fontsize', 8)
text(gridx5(end), gridy5(1), '119','Fontsize', 8)
text(gridx5(1), gridy5(end)/4, '19','Fontsize', 8)
text(gridx5(1), gridy5(end)/2, '39','Fontsize', 8)
text(gridx5(1), gridy5(end)*3/4, '58','Fontsize', 8)
text(gridx5(1), gridy5(end), '78','Fontsize', 8)
text(gridx5(1), gridy5(end)*5/8, '49', 'Fontsize', 8)
text(gridx5(end)-888, gridy5(end)+20, 'Total number = 9401')
hold off


%%
%%% PHYSICALLY TRANSCRIBING X and Y DIMENSION OF GRID POINT
%%% DIMENSIONS - 3, 5, 8, 11,11,11, 11 (0 at 11), 12 (0 at 11, 12), 10
%%%%%%%%%%%%% --------  % ------------------- %
VelxiCor1 = [53, 52, 51,     53, 52, 51, 50, 48,     55, 54, 52, 51, 50,
49, 48, 46,     57, 56, 54, 53, 52, 51, 49, 48, 47, 46, 46,      57, 55,
54, 52, 51, 50, 49, 48, 47, 46, 45,     59, 58, 56, 54, 53, 51, 50, 49, 48,
47, 46                 ,     60, 58, 56, 55, 53, 52, 51, 50, 49, 48, 46,
61, 59, 57, 56, 54, 53, 52, 51, 50, 49, 48
63, 61, 59, 57, 55, 54, 52, 51, 50, 49, 0, 0,         65, 63, 61, 59, 57,
55, 54, 53, 0, 0,                     65,62,60,58,56,55,0,0,0,
65,63,61,59,57,0,0,0,0,       62,60,0,0, 0, 0, 0, 0, 0] ;
VelyiCor1 = [26, 30, 34,     31, 35, 39,42,45,
31,36,40,43,48,51,54,57,            32,36,41,44,48,51,55,58,61,64,66
,37, 41, 45, 48, 51,54,57,60,63, 66, 69,       38, 42, 46, 49, 52, 55, 57,
59, 61, 63, 65,                           43, 47, 51, 54, 57, 60, 63, 65,
67, 69, 71,              48, 52, 56, 59, 62, 65, 67, 69, 71, 73, 75,
```

```
50, 55, 59, 63, 67, 70, 73, 76, 78, 80, 0, 0,          59, 64, 68, 71, 74,
77, 80, 82, 83, 0,            71, 74, 77, 79, 81, 83, 85, 0,0,      77, 80,
83, 87, 90, 0, 0,0,0,        92, 94, 0,0,0,0,0,0,0]     ;
VelXYCor1 = VelxiCor1 .* VelyiCor1; % This are the index locations for our
Validx and Valid y for Batch One

%%%%% BATCH TWO - Sample points to realworld points - Should be the same
%%%%% amount for every batch, just different locations in pixel space
VelxiCor2 = [10,15,20,      16,21,25,30,34,      16,21,26,31,35,40,43,47,
17,22,27,32,36,41,44,48,52,55,59,       23,29,34,39,43,47,51,55,59,62,65,
24,30,35,41,45,49,53,58,61,64,68,       32,36,43,48,51,55,60,63,67,70,0,
33,39,45,50,55,59,63,67,71,74,0,0,      41,47,52,57,62,66,70,73,76,0,
50,56,61,65,70,74,77,81,0,0,    65,70,74,78,82,85,0,0,0,
75,79,83,87,90,0,0,0,0,     93,96,0,0,0,0,     0,0,0];
VelyiCor2 = [36,35,34,      37,36,35,34,33,      39,38,37,36,35,34,33,32,
42,40,39,38,36,35,34,33,32,31,30,       43,42,40,39,37,36,35,34,33,32,31,
47,45,44,42,41,40,39,38,37,36,35,       49,47,45,44,42,40,39,38,37,35,34,
53,51,49,47,45,43,41,40,39,38,0,0,      55,52,50,48,46,44,41,40,39,0,
57,54,52,49,47,45,44,42,0,0,    56,53,51,49,47,45,0,0,0,
58,55,53,50,48,0,0,0,0,     54,52,0,0,0,0,     0,0,0];
VelXYCor2 = VelxiCor2 .* VelyiCor2; % Indexed location of each sample point
as a gridpoint

%%%% BATCH THREE - Sample points to realworld points - Should be the same
%%%%% amount for every batch, just different locations in pixel space
VelxiCor3 = [11,16,21,   16,21,26,31,35,   17,22,27,32,36,40,44,48,
18,23,28,33,38,42,46,50,53,57,60,   24,29,34,39,44,48,52,56,59,62,65,
25,31,36,41,46,50,54,59,63,66,69,       32,38,43,48,53,57,61,65,68,71,0,
34,40,46,51,55,59,63,67,70,74,0,0,   43,48,54,58,63,67,71,75,78,0,
52,57,62,67,71,75,79,82,0,0,    66,70,75,80,83,86,0,0,0,
0,0,85,88,91,0,0,0,0,       0,0,0,0,0,0,     0,0,0];
VelyiCor3 = [37,36,35,   38,37,36,35,33,   40,39,38,37,35,34,33,32,
43,41,40,39,37,36,35,34,33,32,31,   44,42,41,40,39,37,36,35,34,33,32,
47,45,44,42,41,39,38,37,36,34,33,       49,47,45,43,42,40,39,38,36,35,0,
53,50,48,46,44,43,41,40,39,38,0,0,   54,52,50,48,46,44,43,41,40,0,
55,54,51,49,47,45,44,43,0,0,    54,53,51,48,46,45,0,0,0,
0,0,52,50,48,0,0,0,0,       0,0,0,0,0,0,     0,0,0];
VelXYCor3 = VelxiCor3 .* VelyiCor3;

%%%% BATCH FOUR - Sample points to realworld points - Should be the same
%%%%% amount for every batch, just different locations in pixel space
VelxiCor4 = [10,16,21,  16,21,26,30,35,    17,22,27,32,36,40,44,48,
17,23,28,33,37,42,46,50,53,57,60,    24,29,34,39,44,48,52,56,59,62,65,
25,31,36,41,46,50,54,58,62,65,68,       32,38,43,48,53,57,61,65,68,71,0,
34,40,46,51,55,59,64,68,71,74,0,0,   0,48,54,59,63,68,71,75,78,0,
0,57,62,67,71,75,79,0,0,0,     0,45,49,53,56,59,0,0,0,
0,0,57,60,63,0,0,0,0,       0,0,0,0,0,0,     0,0,0];
VelyiCor4 = [35,34,33,  37,35,34,33,32,    39,38,36,35,34,33,32,31,
41,40,39,37,36,35,34,33,32,31,30,    42,40,39,37,36,35,34,33,32,31,30,
45,43,41,40,38,37,36,34,33,32,31,       47,45,43,41,39,38,37,35,34,33,0,
50,48,46,44,42,41,40,39,37,36,0,0,   0,50,48,46,44,42,40,39,38,0,
0,51,49,47,45,43,41,0,0,0,     0,51,48,46,44,43,0,0,0,
0,0,47,45,43,0,0,0,0,       0,0,0,0,0,0,     0,0,0];
VelXYCor4 = VelxiCor4 .* VelyiCor4;

%%%% BATCH FOUR - Sample points to realworld points - Should be the same
%%%%% amount for every batch, just different locations in pixel space
VelxiCor5 = [10,16,21,   16,21,26,30,35,    17,22,27,32,36,40,44,48,
17,23,28,33,38,42,46,50,53,56,59,       24,29,34,39,44,48,52,56,59,62,65,
25,30,36,41,56,60,64,68,71,73,76,     31,37,42,47,53,57,61,64,68,71,0,
33,39,44,50,54,59,64,68,71,74,0,0,      41,46,52,56,71,75,79,83,86,0,
```

```
0,56,61,66,70,74,78,0,0,0,    0,70,75,79,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,      0,0,0];
VelyiCor5 = [35,34,33,   36,35,34,33,32,    39,37,36,35,34,33,32,31,
41,40,39,37,36,35,34,33,32,31,30,      42,40,39,37,36,35,34,33,32,31,31,
45,43,41,40,38,37,36,35,34,33,32,     47,45,43,41,39,38,37,35,34,33,0,
52,50,48,46,44,43,41,40,38,37,0,0,       53,51,49,47,45,43,41,40,38,0,
0,52,50,48,46,44,43,0,0,0,    0,52,49,47,0,0,0,0,0,  0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,      0,0,0];
VelXYCor5 = VelxiCor5 .* VelyiCor5;

%%%% ALL BATCHES DONE %%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% BATCH ONE - ATTAIN VELOCITY SCRIPT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Ascertain the Velocity for X at each Sample Point
VelXYCor1(VelXYCor1==0) = 1; % Convert 0's to all entries of correlation at
grid 1 point

for i = 1:length(VelXYCor1)
VelFinalX1{i} = validx(VelXYCor1(i),:); % Create a cell and save x
positions at every timestep
end
ix = find(VelXYCor1 == 1) % Create logical index where it is equal to 1 to
remove later on.
%%% Ascertain the Velocity for Y at each Sample Point
for i = 1:length(VelXYCor1)
VelFinalY1{i} = validy(VelXYCor1(i),:); % Create a cell and save x
positions at every timestep
end
%%% Attain Magnitude of Change at each Sample Point
%%% Magnitude is sqrt(x^2+y^2)
for i = 1:120
VelFinalMag1{i} =
sqrt((VelFinalX1{i}.*VelFinalX1{i})+(VelFinalY1{i}.*VelFinalY1{i}));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% BATCH TWO - ATTAIN VELOCITY SCRIPT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Ascertain the Velocity for X at each Sample Point
VelXYCor2(VelXYCor2==0) = 1; % Convert 0's to all entries of correlation at
grid 1 for CONTROL

%%% ASSIMILATE ALL VALID DATAS for BATCH TWO
% load ('valid_data.mat')
%% 2A %%
% validx2a = validx;
% validy2a = validy;
% corrcoeff2a = corr_coeff_full;
%% 2B %%
% validx2b = validx;
% validy2b = validy;
% corrcoeff2b = corr_coeff_full;
%% 2C %%
%validx2c = validx;
%validy2c = validy;
%corrcoeff2c = corr_coeff_full;

% validx2All = horzcat(validx2a, validx2b) % 28 entries in TOTAL
% validx2All = horzcat(validx2All, validx2c) % Connect all correlation data
(matrix addition, as each result is a 9282 dimension matrix)
```

```
% validy2All = horzcat(validy2a, validy2b); % Achieved this using horzcat
function (vertical concatenation)
% validy2All = horzcat(validy2All, validy2c); % 28 correlation entries in
TOTAL for this batch

for i = 1:length(VelXYCor2)
VelFinalX2{i} = validx2All(VelXYCor2(i),:); % Create a cell and save x
positions at every timestep
end
VelOutsideInd2 = find(VelXYCor2 == 1) % Create logical index where it is
equal to 0 or 1 to remove later on.
%%% Ascertain the Velocity for Y at each Sample Point
for i = 1:length(VelXYCor2)
VelFinalY2{i} = validy2All(VelXYCor2(i),:); % Create a cell and save x
positions at every timestep
end
%%% Attain Magnitude of Change at each Sample Point
%%% Magnitude is sqrt(x^2+y^2)
for i = length(VelFinalX2)
VelFinalMag2{i} =
sqrt((VelFinalX2{i}.*VelFinalX2{i})+(VelFinalY2{i}.*VelFinalY2{i})); %
Final magnitude of displacement in meters
end
%%% ALL DATA FROM BATCH TWO STORED %%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% BATCH THREE - ATTAIN VELOCITY SCRIPT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Ascertain the Velocity for X at each Sample Point
VelXYCor3(VelXYCor3==0) = 1; % Convert 0's to all entries of correlation at
grid 1 for CONTROL

%%% ASSIMILATE VALID DATA BATCH THREE
% load ('valid_data.mat') % Navigate to 3
% validx3 = validx;
% validy3 = validy;
% corrcoeff3 = corr_coeff_full;

for i = 1:length(VelXYCor3)
VelFinalX3{i} = validx3(VelXYCor3(i),:); % Create a cell and save x
positions at every timestep
end
VelOutsideInd3 = find(VelXYCor3 == 1) % Create logical index where it is
equal to 1 to remove later on.
%%% Ascertain the Velocity for Y at each Sample Point
for i = 1:length(VelXYCor3)
VelFinalY3{i} = validy3(VelXYCor3(i),:); % Create a cell and save x
positions at every timestep
end
%%% Attain Magnitude of Change at each Sample Point
%%% Magnitude is sqrt(x^2+y^2)
for i = 1:length(VelXYCor3)
VelFinalMag3{i} =
sqrt((VelFinalX3{i}.*VelFinalX3{i})+(VelFinalY3{i}.*VelFinalY3{i}));
end

%%% SAVED AND STORED BATCH THREE CORRELATIONS %%%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% BATCH FOUR - ATTAIN VELOCITY SCRIPT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Ascertain the Velocity for X at each Sample Point
```

```
VelXYCor4(VelXYCor4==0) = 1; % Convert 0's to all entries of correlation at
grid 1 for CONTROL

%%% ASSIMILATE VALID DATA BATCH THREE
% load ('valid_data.mat') % Navigate to 3
% validx4a = validx;
% validy4a = validy;
% corrcoeff4a = corr_coeff_full;
% validx4b = validx;
% validy4b = validy;
% corrcoeff4b = corr_coeff_full;
% validx4c = validx;
% validy4c = validy;
% corrcoeff4c = corr_coeff_full;
% validx4d = validx;
% validy4d = validy;
% corrcoeff4d = corr_coeff_full;
% validx4e = validx;
% validy4e = validy;
% corrcoeff4e = corr_coeff_full;
% validx4f = validx;
% validy4f = validy;
% corrcoeff4f = corr_coeff_full;
% validx4g = validx;
% validy4g = validy;
% corrcoeff4g = corr_coeff_full;
% validx4h = validx;
% validy4h = validy;
% corrcoeff4h = corr_coeff_full;

% validx4All = horzcat(validx4a, validx4b, validx4c, validx4d, validx4e,
validx4f, validx4g,validx4h ); % 100 entries in TOTAL
% validy4All = horzcat(validy4a, validy4b, validy4c, validy4d, validy4e,
validy4f, validy4g, validy4h); % 100 Achieved this using horzcat function
(vertical concatenation)

for i = 1:length(VelXYCor4)
VelFinalX4{i} = validx4All(VelXYCor4(i),:); % Create a cell and save x
positions at every timestep
end
VelOutsideInd4 = find(VelXYCor4 == 1) % Create logical index where it is
equal to 1 to remove later on.
%%% Ascertain the Velocity for Y at each Sample Point
for i = 1:length(VelXYCor4)
VelFinalY4{i} = validy4All(VelXYCor4(i),:); % Create a cell and save x
positions at every timestep
end
%%% Attain Magnitude of Change at each Sample Point
%%% Magnitude is sqrt(x^2+y^2)
for i = 1:length(VelXYCor4)
VelFinalMag4{i} =
sqrt((VelFinalX4{i}.*VelFinalX4{i})+(VelFinalY4{i}.*VelFinalY4{i})); % USE
THIS OUTPUT TO FOR NEXT PART USING GSF/PX SIZE
end

%%% SAVED AND STORED FOR BATCH FOUR %%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% BATCH FIVE - ATTAIN VELOCITY SCRIPT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Ascertain the Velocity for X at each Sample Point
```

```
VelXYCor5(VelXYCor5==0) = 1; % Convert 0's to all entries of correlation at
grid 1 for CONTROL

%%% ASSIMILATE ALL VALID DATAS for BATCH TWO
% load ('valid_data.mat')
%% 5A %%
% validx5a = validx;
% validy5a = validy;
% corrcoeff5a = corr_coeff_full;
%% 5B %%
% validx5b = validx;
% validy5b = validy;
% corrcoeff5b = corr_coeff_full;
%% 5C %%
% validx5c = validx;
% validy5c = validy;
% corrcoeff5c = corr_coeff_full;

% validx5All = horzcat(validx5a, validx5b, validx5c) % 28 entries in TOTAL
% validy5All = horzcat(validy5a, validy5b, validy5c); % Achieved this using
horzcat function (vertical concatenation)

for i = 1:length(VelXYCor5)
VelFinalX5{i} = validx5All(VelXYCor5(i),:); % Create a cell and save x
positions at every timestep
end
VelOutsideInd5 = find(VelXYCor5 == 1) % Create logical index where it is
equal to 0 or 1 to remove later on.
%%% Ascertain the Velocity for Y at each Sample Point
for i = 1:length(VelXYCor5)
VelFinalY5{i} = validy5All(VelXYCor5(i),:); % Create a cell and save x
positions at every timestep
end
%%% Attain Magnitude of Change at each Sample Point
%%% Magnitude is sqrt(x^2+y^2)
for i = 1:length(VelFinalX5)
VelFinalMag5{i} =
sqrt((VelFinalX5{i}.*VelFinalX5{i})+(VelFinalY5{i}.*VelFinalY5{i})); %
Final magnitude of displacement in meters
end
%%% ALL DATA FOR BATCH FIVE SAVED %%%

%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% PLOT ALL TIMESTEPS AND ALL MAGNITUDES with GSF
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% BATCH ONE %%%
%%%%%%%%%%%%%%%%%
%%% REMOVE BAD PLOTS
% for k = 1:length(ix)
% VelFinalMag1{ix(k)} = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]; % Create
a 0 vector with dimension 1x20
% end
%%% LOOP TO SUBTRACT THE PRINCIPAL POSITION VECTOR FROM EACH CELL ENTRY
for j = 1:120 % Sample entries
    %VelFinalMag1{j} = VelFinalMag1{j}.*VelGSF(j).*VelPXF(j); % GSF FACTOR
AND
%PIXEL SCALE - Dont Rerun as it will apply more than once.
    for k = 1:20 % Time
        if VelFinalMag1{j} == 0;
           VelFinalMagRaw1{j} = 0;
```

```
            continue
        end
    VelFinalMagRaw1{j}(k) = VelFinalMag1{j}(k) - VelFinalMag1{j}(1); %
Calculates each discrete positional change for the sample points. Output
are displacements in Meters in real world.
    end
end

%%
%%% CUMULATIVE BATCH ONE DATA %%%
%%% NO NEED AS ALL RESULTS ARE TOGETHER BUT FIND THE END RESULT
VelFinalMagRaw1{1} = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]; % Converts
First Entry to 1x20 of 0s
for i = 1:length(ix)
VelFinalMagRaw1{ix(i)} = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]; %
Converts all 0s to 1x20 of 0s
end
%% Remove Last Two Entries of Batch One - as Correlation Coefficient was
very bad...
%%% ERROR FILTERING AND REFINEMENT STEP
for k = 1:18
for j = 1:120
VelFinalMagRaw1Cum{j}(k) = VelFinalMagRaw1{j}(k); % Stores all values
except from 19:20
end
end

for k = 18 % Rather than k = 20
    for j = 1:119
    VelFinalMagRaw1End{j}(k) = VelFinalMagRaw1{j}(k); % Calculates the END
VALUE of BATCH ONE
    for k = 1:17
        VelFinalMagRaw1End{j}(k) = VelFinalMagRaw1End{j}(18); % Assigns the
END VALUE OF BATCH ONE TO EVERY OTHER CELL LOCATION
    end
    end
end
%%% DONE FOR BATCH ONE %%% SAVED AS A 1x119 CELL rather than 1x120 for
%%% consistency with Batches 2,3,4,5
%% FILTERING THE BAD ENTRIES (i.e. the VelOutSideInd)
%%%% NO NEED TO FILTER ANY AS NONE WERE NaNs to start off with. All Points
%%%% Lie on the Glacier?
%%
%%% PLOTS
%%% Create a Colormap
%%% RED TONE
% cmap_out = cmap([1 0 0], 40); %%% RED
% cmap_white = cmap([1 1 1], 40); %%% WHITE
% cmap_orange = cmap([1 0.64], 40); %%% ORANGE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% SOLUTION TO DATA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PROCESSING
%%% FIGURE OUT THE LAST ENTRY OF ALL 120 POINTS
%%% DOING THIS WILL FIND THE FINAL POSITION OF THE CORRELATIONS - MUST DO
%%% THIS BECAUSE IN > BATCH TWO, THERE ARE DIFFERENT SECTIONS, SO IN ORDER
%%% TO FIND THE CUMULATIVE POSITION WE MUST ADD IT TO THE LAST ENTRY.

%VelDate1Final = Datenm1(1:18); % REMOVE LAST TWO BAD CORRELATIONS FROM B1
figure (2)
tic
hold on
%%% PLOT FIRST 40 SAMPLES (furthest back)
```

```
for k = 1:39
    clear cmap_out
%cmap_green = cmap([0.4 1 0.4], 11); %%%
p = plot(VelDate1Final, VelFinalMagRaw1CumNew{k},'g');
hold on
datetick('x')
end
%%% PLOT NEXT 40 SAMPLES (further back)
for j = 40:79
%cmap_bl = cmap([0 0.74 1], 10); %%% BLUE
p2 = plot(VelDate1Final, VelFinalMagRaw1CumNew{j},'b');
hold on
datetick('x')
end
%%% PLOT FINAL 40 SAMPLES (near terminus)
for j = 80:119
%cmap_red = cmap([1 0.6 0.47], 10); %%% RED
p3 = plot(VelDate1Final, VelFinalMagRaw1CumNew{j},'r');
hold on
datetick('x')
end
%%% PLOT INDIVIDUAL PLOTS (i.e. Controls) WITH DIFFERENT COLOURS
for h = 1:length(VelOutsideInd1)
    for j = VelOutsideInd1(h)
    p4 = plot(VelDate1Final, VelFinalMagRaw1Cum{j}(1:18),'c'); % Make sure
its the same length at k position
    hold on
    end
end

axis normal
title ('Batch One Cumulative Positions')
xlabel ('Months of 2013')
ylabel ('Magnitude of Displacements (m)')
legend([p(order) p2(order) p3(order) p4(order)],{'Green Samples', 'Blue
Samples', 'Red Samples', 'Camera Movement (Control)'},'Location',
'NorthEastOutside')

hold off
toc
%%%% ISSUES WITH MATRIX CONCAT in LEGEND CREATION (Solved below)
VelFinalMagRaw1CumNew = VelFinalMagRaw1Cum;
VelOutsideInd1 = ix(1:end-1);% Has an index for 120. which we dont need
for k = 1:length(VelOutsideInd1)
VelFinalMagRaw1CumNew{VelOutsideInd1(k)} = 0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% COMPARING SAMPLES BASED ON EACH GRID TRANSECT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% FIRST ROW - Lots of noise? Why is both +/-
figure (3)
plot(Datenm1, VelFinalMagRaw1{1},'r') % NaN
plot(Datenm1, VelFinalMagRaw1{4},'r')
hold on
plot(Datenm1, VelFinalMagRaw1{2},'r')
plot(Datenm1, VelFinalMagRaw1{3},'r')
plot(Datenm1, VelFinalMagRaw1{5},'g')
datetick ('x')

%%% Comparing Longitudinal profiles %%% Summary of this shows that there
%%% are markedly different velocities for the terminus and further back in
```

```
%%% the terminus

figure (4)
plot(Datenm1(1:18), VelFinalMagRaw1{13}(1:18),'g')
%plot(Datenm1, VelFinalMagRaw1{14},'g') %%% BACK FLOW ON 14
hold on
plot(Datenm1(1:18), VelFinalMagRaw1{12}(1:18),'g')
plot(Datenm1(1:18), VelFinalMagRaw1{3}(1:18),'g')
plot(Datenm1(1:18), VelFinalMagRaw1{5}(1:18),'g')
plot(Datenm1(1:18), VelFinalMagRaw1{45}(1:18),'b')
plot(Datenm1(1:18), VelFinalMagRaw1{43}(1:18),'b')
plot(Datenm1(1:18), VelFinalMagRaw1{41}(1:18),'b')
plot(Datenm1(1:18), VelFinalMagRaw1{89}(1:18),'r')
plot(Datenm1(1:18), VelFinalMagRaw1{90}(1:18),'r')
plot(Datenm1(1:18), VelFinalMagRaw1{98}(1:18),'r')
plot(Datenm1(1:18), VelFinalMagRaw1{96}(1:18),'r')


plot(Datenm1(1:18), VelFinalMagRaw1{94}(1:18),'r') % 97 is short
plot(Datenm1(1:18), VelFinalMagRaw1{91}(1:18),'r') % 92, 93 = 0
datetick ('x')
title ('Position change from sample points in Batch One')
ylabel('Displacement (m)')
xlabel('Months of 2013')




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%  BATCH TWO PLOTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% REMOVE BAD PLOTS
for k = 1:length(VelOutsideInd2)
VelFinalMag2{VelOutsideInd2(k)} = 0;
end
%%% LOOP TO SUBTRACT THE PRINCIPAL POSITION VECTOR FROM EACH CELL ENTRY
for j = 1:119 % Sample entries
%VelFinalMag2{j} = VelFinalMag2{j}.*geom_scale2(j).*px_size2(j); % GSF
FACTOR AND
%PIXEL SCALE imported as their default name geom_scale and px_size2
    for k = 1:28 % Time
        if VelFinalMag2{j} == 0;
           VelFinalMagRaw2{j} = 0;
           continue
        end
    VelFinalMagRaw2{j}(k) = VelFinalMag2{j}(k) - VelFinalMag2{j}(1); %
Calculates each discrete positional change for the sample points. Output
are displacements in Meters in real world.
    end
end
%%% SAVED FINAL GSF AND PXSIZE ALTERED RAW DISPLACEMENTS FOR BATCH TWO %%%
%% CONVERTS ALL 0s to 1x28 of 0s
%%% REMOVE BAD PLOTS
for k = 1:length(VelOutsideInd2)
VelFinalMagRaw2{VelOutsideInd2(k)} =
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]; % Create a 0
vector with dimension 1x28
end
%%% DO NOT RERUN ABOVE %%%%
%% BATCH TWO CUMULATIVE CALCULATIONS %%%
```

```
%%% CONVERT ALL NANs to 0s first for Batch Two
VelFinalMagRaw2NaN =
cellfun(@nanzero,VelFinalMagRaw2,'UniformOutput',false);
VelFinalMagRaw2Cum = VelFinalMagRaw2NaN; % Preallocate it for size

% Assign Final MagRawEnd - DONT RERUN AFTER DOING IT ONCE!
% for k = 1:9 %
%     for j = 1:119
%     VelFinalMagRaw2Cum{j}(k) = VelFinalMagRaw1End{j}(18) +
VelFinalMagRaw2Cum{j}(k) % Calculates the END VALUE of BATCH TWO - MAKE
SURE YOU USE THE CUMULATIVE VALUES
%     end
% end

%%%%% ADD BATCH TWOa (0-9) END VALUES TO BATCH TWOb values (10-16) and
%%%%% likewise BATCH TWOb value (16) to BATCH TWOc Values (17-28)

% WORKS PERFECTLY - Dont rerun!
% for j = 1:119
%     VelFinalMagRaw2Cum{j}(10:16) =
VelFinalMagRaw2Cum{j}(9)+VelFinalMagRaw2Cum{j}(10:16) % THIS MAKES ALL
THOSE THAT HAVE NaNs in them, dissapear. Good Filter?
% end
% WORKS PERFECTLY - Dont rerun!
% for j = 1:119
%     VelFinalMagRaw2Cum{j}(17:28) = VelFinalMagRaw2Cum{j}(16) +
VelFinalMagRaw2Cum{j}(17:28)
% end

%% FILTERING THE BAD ENTRIES (i.e. the VelOutSideInd)
%%%% REMOVE ALL THE 0 ENTRIES COMPLETELY AFTER EVERYTHING HAS BEEN

% for k = 1:length(VelOutsideInd2)
% VelFinalMagRaw2Cum{VelOutsideInd2(k)} =
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]; % Reconvert 0's
array back
% %to NaN %% OR CONVERT BACK IN CASE I NEED TO PLOT ALL
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
% end


%%
%%% ATTAIN TIME STEPS (i.e. Each image date independent of DatenmAll OR our
%%% Positional Dates) We must do this because we introduced more images for
%%% the DIC than the Positional Vector analysis
% load('filenamelist.mat') % Navigate to folder, load the namelist
% VelDate2a = filenamelist % Store the respective namelist
% VelDate2b = filenamelist
% VelDate2c = filenamelist
% VelDate2All = vertcat(VelDate2a, VelDate2b) % Vertical Concatenate the
first two sections of Batch Two of datenames
% VelDate2All = vertcat(VelDate2All, VelDate2c) % Vertical Concatenate the
final section of datenames
VelDate2All = cellstr(VelDate2All);
VelDate2All = cellfun(@(x) x(1:8),VelDate2All(cellfun('length',VelDate2All)
> 7),'un',0); % Cell Fun operation only works on CELLS
for i = 1:length(VelDate2All);
    datecount = VelDate2All(i); % Need to convert the DATE to a STRING
first using datestr
    VelDate2Final(i) = datenum(cellfun(@(x) {[x(1:4) '-' x(5:6) '-'
x(7:8)]}, VelDate2All(i)), 29); % Convert Datetime to Datenum because
errorbar function doesn't like datetime formats
```

```
end

%%
figure (6)
tic
hold on
%%% PLOT FIRST 40 SAMPLES (furthest back)
for k = 1:39
    clear cmap_out
%cmap_green = cmap([0.4 1 0.4], 11); %%%
p = plot(VelDate2Final, VelFinalMagRaw2CumNew{k},'g');
hold on
datetick('x')
end
%%% PLOT NEXT 40 SAMPLES (further back)
for j = 40:79
%cmap_bl = cmap([0 0.74 1], 10); %%% BLUE
p2 = plot(VelDate2Final, VelFinalMagRaw2CumNew{j},'b');
hold on
datetick('x')
end
%%% PLOT FINAL 40 SAMPLES (near terminus)
for j = 80:119
%cmap_red = cmap([1 0.6 0.47], 10); %%% RED
p3 = plot(VelDate2Final, VelFinalMagRaw2CumNew{j},'r');
hold on
datetick('x')
end
%%% PLOT INDIVIDUAL PLOTS (i.e. Controls) WITH DIFFERENT COLOURS
for h = 1:length(VelOutsideInd2)
    for j = VelOutsideInd2(h)
    p4 = plot(VelDate2Final, VelFinalMagRaw2Cum{j}(1:28),'c'); % Make sure
its the same length at k position
    hold on
    end
end
axis normal
title ('Batch Two Cumulative Positions')
xlabel ('Months of 2013')
ylabel ('Magnitude of Displacements (m)')
legend([p(order) p2(order) p3(order) p4(order)],{'Green Samples', 'Blue
Samples', 'Red Samples', 'Camera Movement (Control)'},'Location',
'NorthEastOutside')

hold off
toc

%%% PLOT SPECIFICALLY %%%
%%% SELECTING THE BEST DATA %%%
figure (7)
plot(VelDate2Final, VelFinalMagRaw2{13}(1:28),'g')
%plot(Datenm1, VelFinalMagRaw1{14},'g') %%% BACK FLOW ON 14
hold on
plot(VelDate2Final, VelFinalMagRaw2{12}(1:28),'g')
plot(VelDate2Final, VelFinalMagRaw2{3}(1:28),'g')
plot(VelDate2Final, VelFinalMagRaw2{5}(1:28),'g')
plot(VelDate2Final, VelFinalMagRaw2{45}(1:28),'b')
plot(VelDate2Final, VelFinalMagRaw2{43}(1:28),'b')
plot(VelDate2Final, VelFinalMagRaw2{41}(1:28),'b')
plot(VelDate2Final, VelFinalMagRaw2{89}(1:28),'r')
plot(VelDate2Final, VelFinalMagRaw2{90}(1:28),'r')
```

```
plot(VelDate2Final, VelFinalMagRaw2{98}(1:28),'r')
plot(VelDate2Final, VelFinalMagRaw2{96}(1:28),'r')

plot(VelDate2Final, VelFinalMagRaw2{94}(1:28),'r') % 97 is short
%plot(VelDate2Final, VelFinalMagRaw2{91}(1:28),'r') % 92, 93 = 0
datetick ('x')
title ('Position change from sample points in Batch Two')
ylabel('Displacement (m)')
xlabel('Months of 2013')




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%  BATCH THREE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% LOOP TO SUBTRACT THE PRINCIPAL POSITION VECTOR FROM EACH CELL ENTRY
% for j = 1:119 % Sample entries
% %     VelFinalMag3{j} = VelFinalMag3{j}.*geom_scale3(j).*px_size3(j); %
GSF FACTOR AND
% %PIXEL SCALE
%     for k = 1:16 % Time
%         if VelFinalMag3{j} == 0;
%            VelFinalMagRaw3{j} = 0;
%            continue
%         end
%     VelFinalMagRaw3{j}(k) = VelFinalMag3{j}(k) - VelFinalMag3{j}(1); %
Calculates each discrete positional change for the sample points. Output
are displacements in Meters in real world.
%     end
% end
%% BATCH THREE CUMULATIVE CALCULATIONS %%%
for k = 28 %
    for j = 1:119
    VelFinalMagRaw2End{j}(k) = VelFinalMagRaw2{j}(k); % Calculates the END
VALUE of BATCH ONE
    for k = 1:27
        VelFinalMagRaw2End{j}(k) = VelFinalMagRaw2End{j}(28); % Assigns the
END VALUE OF BATCH ONE TO EVERY OTHER CELL LOCATION
    end
    end
end

for i = 1:length(VelOutsideInd3)
VelFinalMagRaw3{VelOutsideInd3(i)} = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]; %
Converts all 0s to 1x16 of 0s
end
%%%%% ADD BATCH TWO TO BATCH THREE VALUES % N.B. Only apply to first
%%%%% sub-batch to avoid double COUNTING!
% for k = 1:16
%     for j = 1:119
% VelFinalMagRaw3Cum{j}(k) = VelFinalMagRaw2End{j}(28) +
VelFinalMagRaw3{j}(k); %% WORKS - Don't run again.
%     end
% end
%%% FINAL CUMULATIVE CALCULATED and SAVED AS VELFINALMAGRAW3CUM

%%
```

```
%%% ATTAIN TIME STEPS (i.e. Each image date independent of DatenmAll OR our
%%% Positional Dates) We must do this because we introduced more images for
%%% the DIC than the Positional Vector analysis
% load('filenamelist.mat') % Navigate to folder, load the namelist
% VelDate3All = filenamelist % Store the respective namelist

VelDate3All = cellstr(VelDate3All);
VelDate3All = cellfun(@(x) x(1:8),VelDate3All(cellfun('length',VelDate3All)
> 7),'un',0); % Cell Fun operation only works on CELLS
for i = 1:length(VelDate3All);
    datecount = VelDate3All(i); % Need to convert the DATE to a STRING
first using datestr
    VelDate3Final(i) = datenum(cellfun(@(x) {[x(1:4) '-' x(5:6) '-'
x(7:8)]}, VelDate3All(i)), 29); % Convert Datetime to Datenum because
errorbar function doesn't like datetime formats
end

%%%% PLOTTING BATCH THREE %%%% Use VelDate3Final as Time
figure (8)
tic
hold on
%%% PLOT FIRST 40 SAMPLES (furthest back)
for k = 1:39
    clear cmap_out
%cmap_green = cmap([0.4 1 0.4], 11); %%%
p = plot(VelDate3Final, VelFinalMagRaw3CumNew{k},'g');
hold on
datetick('x')
end
%%% PLOT NEXT 40 SAMPLES (further back)
for j = 40:79
%cmap_bl = cmap([0 0.74 1], 10); %%% BLUE
p2 = plot(VelDate3Final, VelFinalMagRaw3CumNew{j},'b');
hold on
datetick('x')
end
%%% PLOT FINAL 40 SAMPLES (near terminus)
for j = 80:119
%cmap_red = cmap([1 0.6 0.47], 10); %%% RED
p3 = plot(VelDate3Final, VelFinalMagRaw3CumNew{j},'r');
hold on
datetick('x')
end
%%% PLOT INDIVIDUAL PLOTS (i.e. Controls) WITH DIFFERENT COLOURS
for h = 1:length(VelOutsideInd3)
    for j = VelOutsideInd3(h)
    p4 = plot(VelDate3Final, VelFinalMagRaw3Cum{j}(1:16),'c'); % Make sure
its the same length at k position
    hold on
    end
end
axis normal
title ('Batch Three Cumulative Positions')
xlabel ('Months of 2013-2014')
ylabel ('Magnitude of Displacements (m)')
legend([p(order) p2(order) p3(order) p4(order)],{'Green Samples', 'Blue
Samples', 'Red Samples', 'Camera Movement (Control)'},'Location',
'NorthEastOutside')

hold off
toc
```

```
%%% PLOT SPECIFICALLY %%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%  BATCH FOUR %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k = 1:length(VelOutsideInd4)
VelFinalMag4{VelOutsideInd4(k)} =
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]; % Create a 0 vector with
dimension 1x20
end
%%% LOOP TO SUBTRACT THE PRINCIPAL POSITION VECTOR FROM EACH CELL ENTRY
% for j = 1:119 % Sample entries
%     VelFinalMag4{j} = VelFinalMag4{j}.*geom_scale4(j).*px_size4(j); % GSF
FACTOR AND
% %PIXEL SCALE
%     for k = 1:100 % Time
%         if VelFinalMag4{j} == 0;
%            VelFinalMagRaw4{j} = 0;
%            continue
%         end
%     VelFinalMagRaw4{j}(k) = VelFinalMag4{j}(k) - VelFinalMag4{j}(1); %
Calculates each discrete positional change for the sample points. Output
are displacements in Meters in real world.
%     end
% end
%% BATCH FOUR CUMULATIVE CALCULATIONS %%%
for i = 1:length(VelOutsideInd4)
VelFinalMagRaw4{VelOutsideInd4(i)} =
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,00,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,00,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,00,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]; % Converts all 0s
to 1x16 of 0s
end
% Assign Final MagRawEnd
for k = 16 % Last entry of BATCH THREE
    for j = 1:119
    VelFinalMagRaw3End{j}(k) = VelFinalMagRaw3Cum{j}(16); % Calculates the
END VALUE of BATCH TWO - MAKE SURE YOU USE THE CUMULATIVE VALUES
%     for k = 1:15 % First to Second Last entry for BATCH TWO
%         VelFinalMagRaw3End{j}(k) = VelFinalMagRaw3End{j}(16); % Assigns
the END VALUE OF BATCH ONE TO EVERY OTHER CELL LOCATION
%     end
    end
end
%%%%% ADD BATCH THREE TO BATCH FOUR VALUES % N.B. Only apply to first
%%%%% sub-batch to avoid double COUNTING!

VelFinalMagRaw4NaN =
cellfun(@nanzero,VelFinalMagRaw4,'UniformOutput',false);
VelFinalMagRaw4Cum = VelFinalMagRaw4NaN;  % Translate it to preallocate
1x100

%% CUMULATIVE CALCULATIONS - 4a first
% for k = 1:10
%     for j = 1:119
%     VelFinalMagRaw4Cum{j}(k) = VelFinalMagRaw3End{j}(16) +
VelFinalMagRaw4NaN{j}(k) ; %% WORKS - Don't run again.
%     end
```

```
% end
% %% 4a has 10 samples, 4b - 3, 4c - 15, 4d - 20, 4e - 19, 4f - 11, 4g -
11,
% %% 4h - 11 samples....
% % % WORKS PERFECTLY - Dont rerun! - 4b
% for j = 1:119
%     VelFinalMagRaw4Cum{j}(11:13) =
VelFinalMagRaw4Cum{j}(10)+VelFinalMagRaw4Cum{j}(11:13) % THIS MAKES ALL
THOSE THAT HAVE NaNs in them, dissapear. Good Filter?
% end
% %WORKS PERFECTLY - Dont rerun! - 4c
% for j = 1:119
%     VelFinalMagRaw4Cum{j}(14:28) = VelFinalMagRaw4Cum{j}(13) +
VelFinalMagRaw4Cum{j}(14:28)
% end
% %WORKS PERFECTLY - Don't rerun! - 4d
% for j = 1:119
%     VelFinalMagRaw4Cum{j}(29:48) = VelFinalMagRaw4Cum{j}(28) +
VelFinalMagRaw4Cum{j}(29:48)
% end
% %WORKS PERFECTLY - Don't rerun! - 4e
% for j = 1:119
%     VelFinalMagRaw4Cum{j}(49:67) = VelFinalMagRaw4Cum{j}(48) +
VelFinalMagRaw4Cum{j}(49:67)
% end
% %WORKS PERFECTLY - Don't rerun! - 4f
% for j = 1:119
%     VelFinalMagRaw4Cum{j}(68:78) = VelFinalMagRaw4Cum{j}(67) +
VelFinalMagRaw4Cum{j}(68:78)
% end
% %WORKS PERFECTLY - Don't rerun! - 4g
% for j = 1:119
%     VelFinalMagRaw4Cum{j}(79:89) = VelFinalMagRaw4Cum{j}(78) +
VelFinalMagRaw4Cum{j}(79:89)
% end
% %WORKS PERFECTLY - Don't rerun! - 4h
% for j = 1:119
%     VelFinalMagRaw4Cum{j}(90:100) = VelFinalMagRaw4Cum{j}(89) +
VelFinalMagRaw4Cum{j}(90:100)
% end
% %% FINAL CUMULATIVE CALCULATED and SAVED AS VELFINALMAGRAW3CUM


%%
%%% ATTAIN TIME STEPS (i.e. Each image date independent of DatenmAll OR our
%%% Positional Dates) We must do this because we introduced more images for
%%% the DIC than the Positional Vector analysis
% load('filenamelist.mat') % Navigate to folder, load the namelist
% VelDate4a = filenamelist % Store the respective namelist
% VelDate4b = filenamelist
% VelDate4c = filenamelist
% VelDate4d = filenamelist
% VelDate4e = filenamelist
% VelDate4f = filenamelist
% VelDate4g = filenamelist
% VelDate4h = filenamelist

% VelDate4All = vertcat(VelDate4a, VelDate4b, VelDate4c, VelDate4d,
VelDate4e, VelDate4f, VelDate4g, VelDate4h) % Vertical Concatenate the
first two sections of Batch Two of datenames
VelDate4All = cellstr(VelDate4All);
```

```
VelDate4All = cellfun(@(x) x(1:8),VelDate4All(cellfun('length',VelDate4All)
> 7),'un',0); % Cell Fun operation only works on CELLS
for i = 1:length(VelDate4All);
    datecount = VelDate4All(i); % Need to convert the DATE to a STRING
first using datestr
    VelDate4Final(i) = datenum(cellfun(@(x) {[x(1:4) '-' x(5:6) '-'
x(7:8)]}, VelDate4All(i)), 29); % Convert Datetime to Datenum because
errorbar function doesn't like datetime formats
end
%%% REMOVE THE BAD NaN Entry at 77-78 of Every Sample
for j = 1:119
VelFinalMagRaw4CumNew{j}(78) = NaN;
VelFinalMagRaw4CumNew{j}(77) = NaN;

end
% N.B. When you run this, make sure to reallocate the VelOutsideInd4 values
% to 0 again in Batch Five....

%%% PLOT BATCH FOUR %%% Use VelDate4Final as Time - Requires
%%% VelFinalMagRaw4CumNew Processed (in Batch Five Chapter of this Code)
figure (10)
tic
hold on
%%% PLOT FIRST 40 SAMPLES (furthest back)
for k = 1:39
    clear cmap_out
%cmap_green = cmap([0.4 1 0.4], 11); %%%
p = plot(VelDate4Final, VelFinalMagRaw4CumNew{k},'g');
hold on
datetick('x')
end
%%% PLOT NEXT 40 SAMPLES (further back)
for j = 40:79
%cmap_bl = cmap([0 0.74 1], 10); %%% BLUE
p2 = plot(VelDate4Final, VelFinalMagRaw4CumNew{j},'b');
hold on
datetick('x')
end
%%% PLOT FINAL 40 SAMPLES (near terminus)
for j = 80:119
%cmap_red = cmap([1 0.6 0.47], 10); %%% RED
p3 = plot(VelDate4Final, VelFinalMagRaw4CumNew{j},'r');
hold on
datetick('x')
end
%%% PLOT INDIVIDUAL PLOTS (i.e. Controls) WITH DIFFERENT COLOURS
for h = 1:length(VelOutsideInd4)
    for j = VelOutsideInd4(h)
    p4 = plot(VelDate4Final, VelFinalMagRaw4Cum{j}(1:100),'c'); % Make sure
its the same length at k position
    hold on
    end
end
axis normal
title ('Batch Four Cumulative Positions')
xlabel ('Quarterly periods of 2014-2015')
ylabel ('Magnitude of Displacements (m)')
legend([p(order) p2(order) p3(order) p4(order)],{'Green Samples', 'Blue
Samples', 'Red Samples', 'Camera Movement (Control)'},'Location',
'NorthEastOutside')
```

```
hold off
toc

%%% PLOT SPECIFICALLY %%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%  BATCH FIVE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% CALCULATE VelFinalMagRaw4End
% for i = 1:length(VelOutsideInd5)
% VelFinalMagRaw5{VelOutsideInd5(i)} =
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]; % Converts all 0s to
1x24 of 0s
% end
% Assign Final MagRawEnd
% for k = 100 % Last entry of BATCH THREE
%     for j = 1:119
%     VelFinalMagRaw4End{j}(k) = VelFinalMagRaw4Cum{j}(100); % Calculates
the END VALUE of BATCH TWO - MAKE SURE YOU USE THE CUMULATIVE VALUES
% %     for k = 1:99 % First to Second Last entry for BATCH TWO
% %         VelFinalMagRaw4End{j}(k) = VelFinalMagRaw4End{j}(100); %
Assigns the END VALUE OF BATCH ONE TO EVERY OTHER CELL LOCATION
% %     end
%     end
% end

for j = 1:119 % Sample entries
    %VelFinalMag5{j} = VelFinalMag5{j}.*geom_scale4(j).*px_size4(j); % GSF
FACTOR AND
%PIXEL SCALE - Dont Rerun as it will apply more than once.
    for k = 1:24 % Time - 24 long for 5th Batch
        if VelFinalMag5{j} == 0;
           VelFinalMagRaw5{j} = 0;
           continue
        end
    VelFinalMagRaw5{j}(k) = VelFinalMag5{j}(k) - VelFinalMag5{j}(1); %
Calculates each discrete positional change for the sample points. Output
are displacements in Meters in real world.
    end
end


%% Create a NaNVector for Batch Five

VelFinalMagRaw5NaN =
cellfun(@nanzero,VelFinalMagRaw5,'UniformOutput',false);
VelFinalMagRaw5Cum = VelFinalMagRaw5NaN;  % Translate it to preallocate
1x100

%% ADD THE RAW END VALUE FOR CUMULATIVE CALCULATIONS
% 5a
for k = 1:9
    for j = 1:119
    VelFinalMagRaw5Cum{j}(k) = VelFinalMagRaw4End{j}(100) +
VelFinalMagRaw5NaN{j}(k) ; %% WORKS - Don't run again.
    end
end
% 5b
for k = 10:12
    for j = 1:119
```

```
    VelFinalMagRaw5Cum{j}(k) = VelFinalMagRaw5Cum{j}(9) +
VelFinalMagRaw5Cum{j}(k) ; %% WORKS - Don't run again.
     end
end
% 5c
for k = 13:24
    for j = 1:119
        VelFinalMagRaw5Cum{j}(k) = VelFinalMagRaw5Cum{j}(12) +
VelFinalMagRaw5Cum{j}(k);
    end
end
%% TIME SERIES for BATCH FIVE
%%% ATTAIN TIME STEPS (i.e. Each image date independent of DatenmAll OR our
%%% Positional Dates) We must do this because we introduced more images for
%%% the DIC than the Positional Vector analysis
%load('filenamelist.mat') % Navigate to folder, load the namelist
%VelDate5a = filenamelist % Store the respective namelist
%VelDate5b = filenamelist
%VelDate5c = filenamelist


% VelDate5All = vertcat(VelDate5a, VelDate5b, VelDate5c)  % Vertical
Concatenate the first two sections of Batch Two of datenames
VelDate5All = cellstr(VelDate5All);
VelDate5All = cellfun(@(x) x(1:8),VelDate5All(cellfun('length',VelDate5All)
> 7),'un',0); % Cell Fun operation only works on CELLS
for i = 1:length(VelDate5All);
    datecount = VelDate5All(i); % Need to convert the DATE to a STRING
first using datestr
    VelDate5Final(i) = datenum(cellfun(@(x) {[x(1:4) '-' x(5:6) '-'
x(7:8)]}, VelDate5All(i)), 29); % Convert Datetime to Datenum because
errorbar function doesn't like datetime formats
end


%% CONTROLS
%%% REMOVE ALL CONTROL VALUES FOR ALL BATCHES
VelFinalMagRaw1CumNew = VelFinalMagRaw1Cum;
for k = 1:length(ix)
VelFinalMagRaw1CumNew{ix(k)} = 0;
end
VelFinalMagRaw2CumNew = VelFinalMagRaw2Cum;
for k = 1:length(VelOutsideInd2)
VelFinalMagRaw2CumNew{VelOutsideInd2(k)} = 0;
end
VelFinalMagRaw3CumNew = VelFinalMagRaw3Cum;
for k = 1:length(VelOutsideInd3)
VelFinalMagRaw3CumNew{VelOutsideInd3(k)} = 0;
end
VelFinalMagRaw4CumNew = VelFinalMagRaw4Cum;
for k = 1:length(VelOutsideInd4)
VelFinalMagRaw4CumNew{VelOutsideInd4(k)} = 0;
end
VelFinalMagRaw5CumNew = VelFinalMagRaw5Cum;
for k = 1:length(VelOutsideInd5)
VelFinalMagRaw5CumNew{VelOutsideInd5(k)} = 0;
end


%% PLOT FOR BATCH FIVE
figure (11)
tic
```

```
hold on
%%% PLOT FIRST 40 SAMPLES (furthest back)
for k = 1:39
    clear cmap_out
%cmap_green = cmap([0.4 1 0.4], 11); %%%
p = plot(VelDate5Final, VelFinalMagRaw5CumNew{k},'g');
hold on
datetick('x')
end
%%% PLOT NEXT 40 SAMPLES (further back)
for j = 40:79
%cmap_bl = cmap([0 0.74 1], 10); %%% BLUE
p2 = plot(VelDate5Final, VelFinalMagRaw5CumNew{j},'b');
hold on
datetick('x')
end
%%% PLOT FINAL 40 SAMPLES (near terminus)
for j = 80:119
%cmap_red = cmap([1 0.6 0.47], 10); %%% RED
p3 = plot(VelDate5Final, VelFinalMagRaw5CumNew{j},'r');
hold on
datetick('x')
end
%%% PLOT INDIVIDUAL PLOTS (i.e. Controls) WITH DIFFERENT COLOURS
for h = 1:length(VelOutsideInd5)
    for j = VelOutsideInd5(h)
    p4 = plot(VelDate5Final, VelFinalMagRaw5Cum{j}(1:24),'c')
    hold on
    end
end

hold off
axis normal
title ('Batch Five Cumulative Positions')
xlabel ('Months of 2015-2016')
ylabel ('Magnitude of Displacements (m)')
legend([p(order) p2(order) p3(order) p4(order)],{'Green Samples', 'Blue
Samples', 'Red Samples', 'Camera Movement (Control)'},'Location',
'NorthEastOutside')
toc
```

```
%% COMBINING ALL TIME SERIES AND ALL BATCHES
%%%%%%%%%%%%%%%

VelDate0All = horzcat(VelDate1Final, VelDate2Final, VelDate3Final,
VelDate4Final, VelDate5Final);
% REDUCE NUMBER OF SAMPLE POINTS TO 119 to be consistent with Batches 2,3,4
for k = 1:18
for j = 1:119
VelFinalMagRaw1Cum{j}(k) = VelFinalMagRaw1{j}(k); % Stores all values
except from 19:20
end
end
%%% Now, Concatenate all the values so that it becomes a 1x XXX within a
%%% 1x199 cell array... not Horzcat
```

```
for j = 1:119
VelFinalMagRaw0Cum{j} = horzcat(VelFinalMagRaw1Cum{j},
VelFinalMagRaw2Cum{j}, VelFinalMagRaw3Cum{j}, VelFinalMagRaw4Cum{j},
VelFinalMagRaw5Cum{j});
end


%%
%%%% PLOT WITH ALL OF THE BATCHES COMBINED!!! 3 years worth of
%%%% displacement.
figure (12)
tic
hold on
%%% PLOT FIRST 40 SAMPLES (furthest back)
for k = 1:39
    clear cmap_out
%cmap_green = cmap([0.4 1 0.4], 11); %%%
plot(VelDate0All, VelFinalMagRaw0Cum{k},'g')
hold on
datetick('x')
end
%%% PLOT NEXT 40 SAMPLES (further back)
for j = 40:79
%cmap_bl = cmap([0 0.74 1], 10); %%% BLUE
plot(VelDate0All, VelFinalMagRaw0Cum{j},'b')
hold on
datetick('x')
end
%%% PLOT FINAL 40 SAMPLES (near terminus)
for j = 80:119
%cmap_red = cmap([1 0.6 0.47], 10); %%% RED
plot(VelDate0All, VelFinalMagRaw0Cum{j},'r')
hold on
datetick('x')
end
hold off
toc

%% COMBINING ALL TIME SERIES AND ALL BATCHES FOR NEW - REMOVING ALL
CONTROLS and REPLOTTING AS CYAN
%%%%%%%%%%%%%%

VelDate0All = horzcat(VelDate1Final, VelDate2Final, VelDate3Final,
VelDate4Final, VelDate5Final);
% REDUCE NUMBER OF SAMPLE POINTS TO 119 to be consistent with Batches 2,3,4
%%% Now, Concatenate all the values so that it becomes a 1x XXX within a
%%% 1x199 cell array... not Horzcat
%%% BECAUSE IM CONCATING FIRST, Then i must have the VECTORS THE SAME SIZE
%%% - Before i remove the Nan Values
for j = 1:119 % NEED TO INCORPORATE THE FIXED NaN in Batch4
VelFinalMagRaw0CumNew1{j} = horzcat(VelFinalMagRaw1Cum{j},
VelFinalMagRaw2Cum{j}, VelFinalMagRaw3Cum{j}, VelFinalMagRaw4CumNew{j},
VelFinalMagRaw5Cum{j});
end

VelFinalMagRaw0CumNew = VelFinalMagRaw0CumNew1;
for k = 1:length(VelOutsideInd5)
VelFinalMagRaw0CumNew{VelOutsideInd5(k)} =
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];
end
%%% WIERD ANOMALY with CELL SIZE corrected in Sample 73
VelFinalMagRaw0CumNew{73}(1:186) =
[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0];

figure (13)
tic
hold on
%%% PLOT FIRST 40 SAMPLES (furthest back)
for k = 1:39
    clear cmap_out
%cmap_green = cmap([0.4 1 0.4], 11); %%%
plot(VelDate0All, VelFinalMagRaw0CumNew{k},'g')
hold on
datetick('x')
end
%%% PLOT NEXT 40 SAMPLES (further back)

for j = 40:79
%cmap_bl = cmap([0 0.74 1], 10); %%% BLUE
plot(VelDate0All, VelFinalMagRaw0CumNew{j},'b')
hold on
datetick('x')
end
%%% PLOT FINAL 40 SAMPLES (near terminus)
for j = 80:119
%cmap_red = cmap([1 0.6 0.47], 10); %%% RED
plot(VelDate0All, VelFinalMagRaw0CumNew{j},'r')
hold on
datetick('x')
end

%%% PLOT INDIVIDUAL PLOTS (i.e. Controls) WITH DIFFERENT COLOURS
for h = 1:length(VelOutsideInd1)
    for j = VelOutsideInd1(h)
    p41 = plot(VelDate1Final, VelFinalMagRaw1Cum{j}(1:18),'c'); % Make sure
its the same length at k position
    hold on
    end
end
for h = 1:length(VelOutsideInd2)
    for j = VelOutsideInd2(h)
    p42 = plot(VelDate2Final, VelFinalMagRaw2Cum{j}(1:28),'c'); % Make sure
its the same length at k position
    hold on
    end
end
for h = 1:length(VelOutsideInd3)
    for j = VelOutsideInd3(h)
    p43 = plot(VelDate3Final, VelFinalMagRaw3Cum{j}(1:16),'c'); % Make sure
its the same length at k position
    hold on
    end
end
for h = 1:length(VelOutsideInd4)
```

```
    for j = VelOutsideInd4(h)
    p44 = plot(VelDate4Final, VelFinalMagRaw4Cum{j}(1:100),'c'); % Make
sure its the same length at k position
    hold on
    end
end
for h = 1:length(VelOutsideInd5)
    for j = VelOutsideInd5(h)
    p45 = plot(VelDate5Final, VelFinalMagRaw5Cum{j}(1:24),'c'); % Make sure
its the same length at k position
    hold on
    end
end

%%% NEED TO MAKE p4 the same size as p, p2, p3 (186 length) before CONCAT %
%%% LAST THING TO FIX IS RIGHT HERE ....
p4 = vertcat(p41, p42, p43, p44, p45);

axis normal
title ('All Batches Cumulative Positions')
xlabel ('2013-2016')
ylabel ('Magnitude of Displacements (m)')
legend([p(order) p2(order) p3(order) p4(order)],{'Green Samples', 'Blue
Samples', 'Red Samples', 'Camera Movement (Control)'},'Location',
'NorthEastOutside')

hold off
toc
```

Translation Outputs from the DirectionOfFlowAttain.m script

Batch One (DIC Grid to real-world sample points)

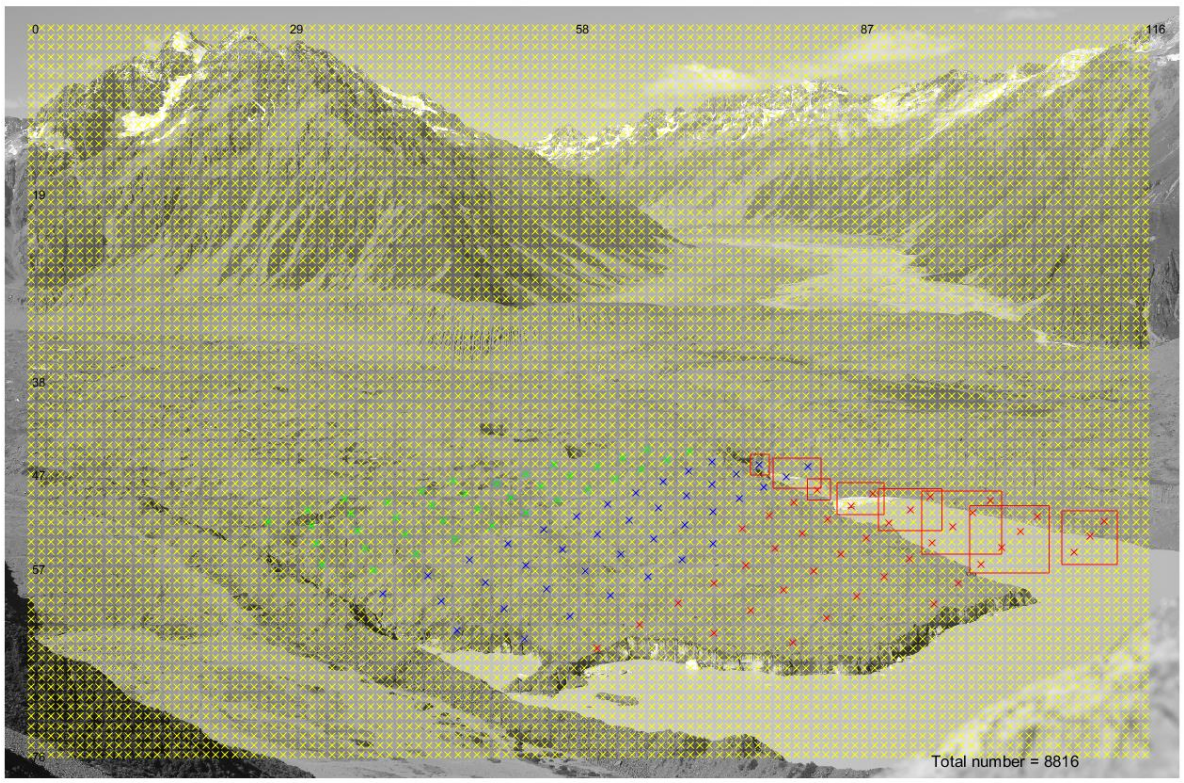


Figure Appendix 2.1. Batch One data translation. Highlighted sample points are removed in the script as correlations at this locations do not reflect changes on the glacier. They were assigned a grid location of 1 to serve as a control for the other samples. Total number of grid points from the DIC algorithm was 8816.

Batch Two (DIC Grid to real-world sample points)

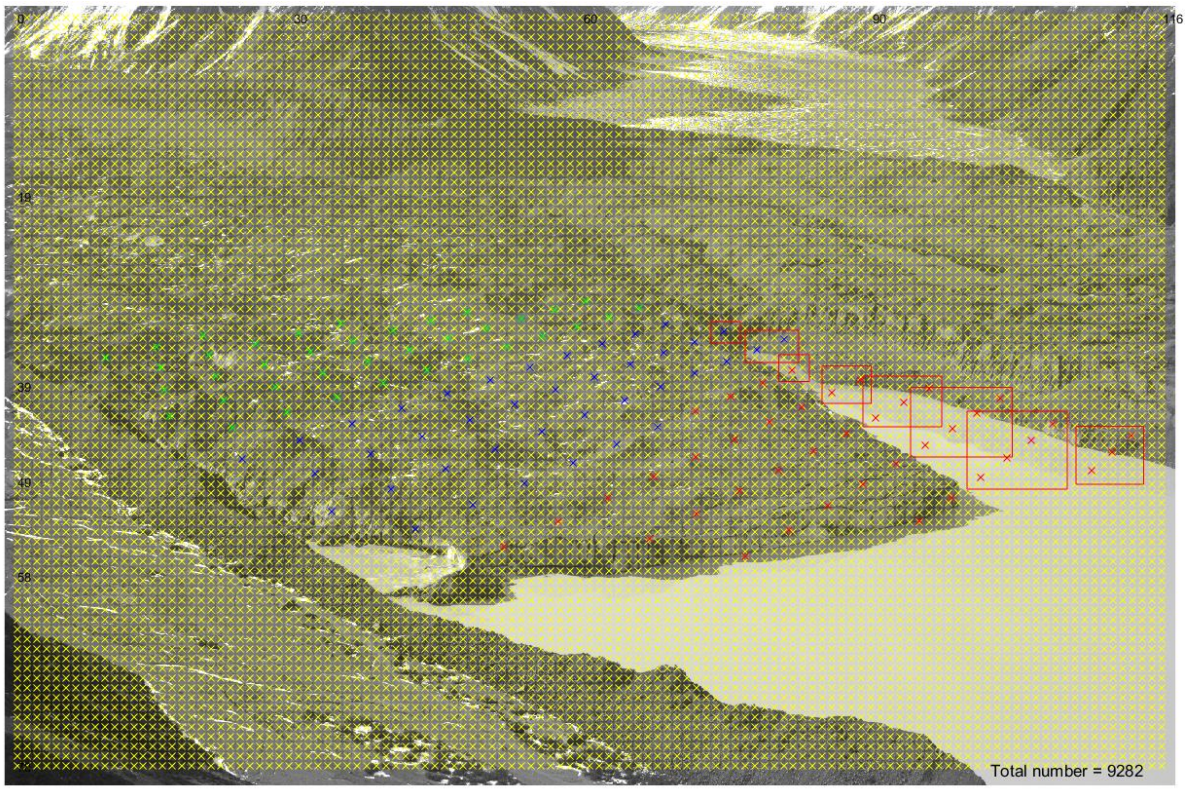


Figure Appendix 2.2. Batch Two data translation. Highlighted sample points are removed in the script as correlations at this locations do not reflect changes on the glacier. They were assigned a grid location of 1 to serve as a control for the other samples. Total number of grid points from the DIC algorithm was 9282.

Batch Three (DIC Grid to real-world sample points)

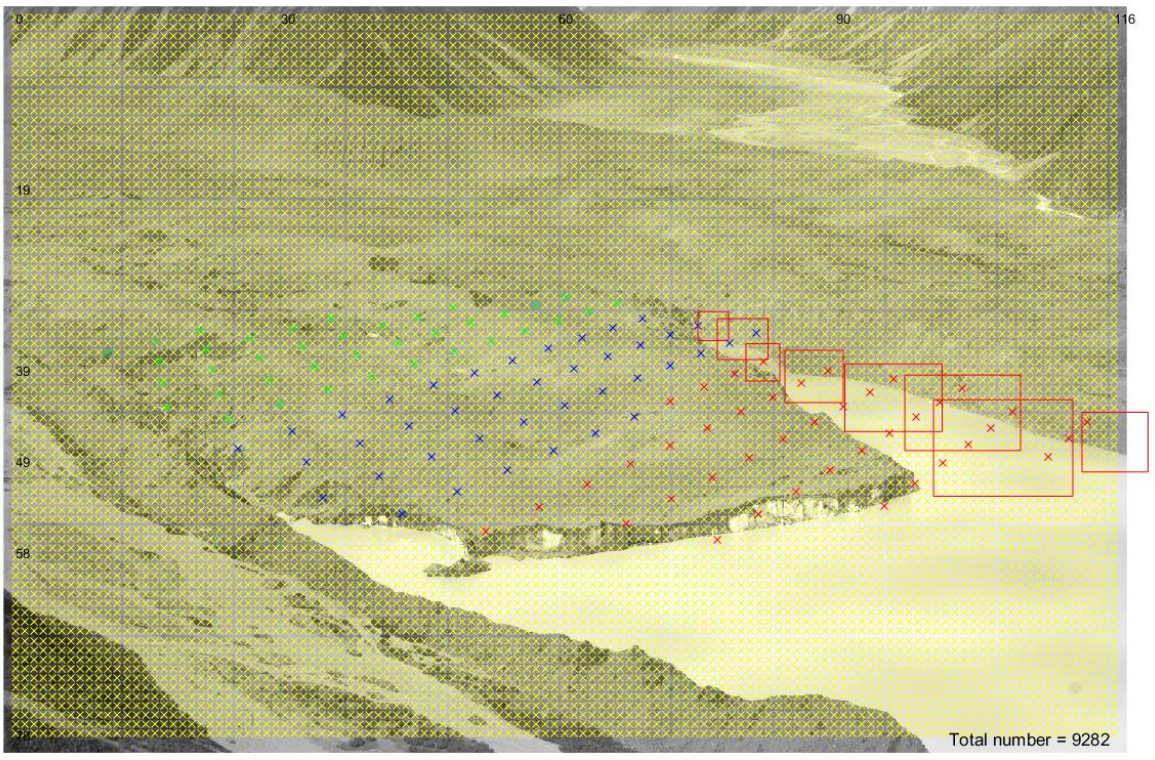


Figure Appendix 2.3. Batch Three data translation. Highlighted sample points are removed in the script as correlations at this locations do not reflect changes on the glacier. They were assigned a grid location of 1 to serve as a control for the other samples. Total number of grid points from the DIC algorithm was 9282.

Batch Four (DIC Grid to real-world sample points)

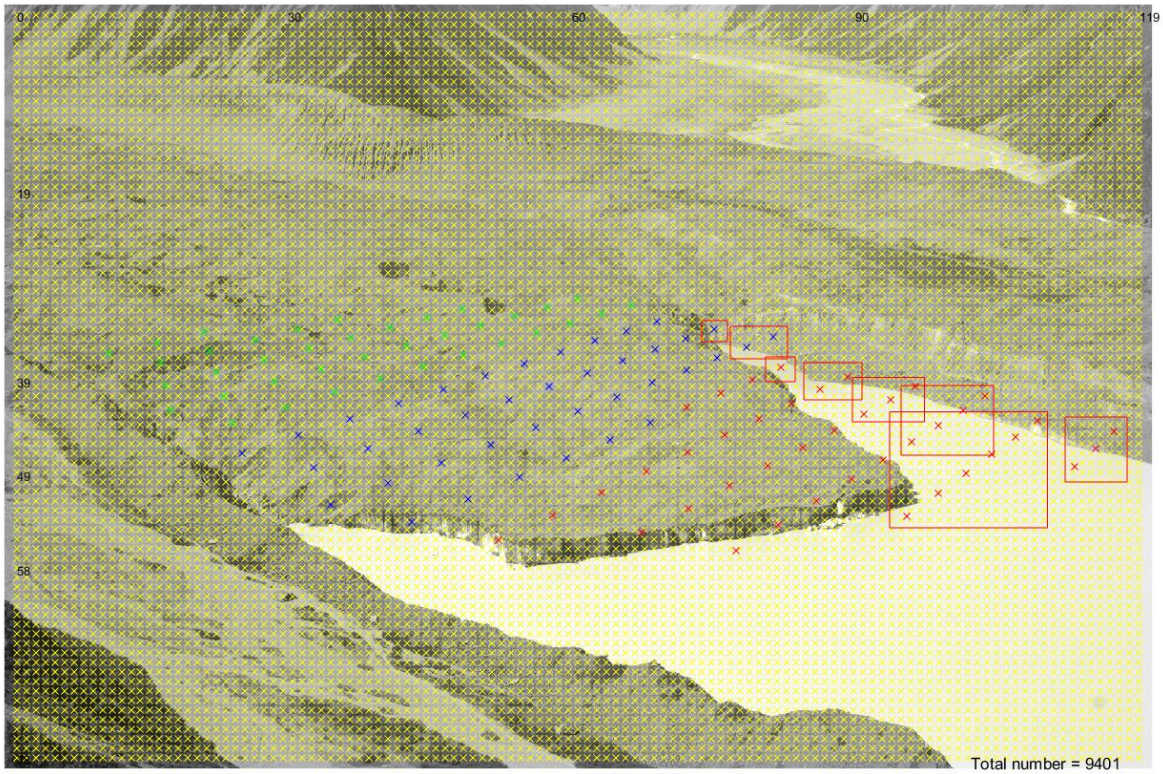


Figure Appendix 2.4. Batch Four data translation. Highlighted sample points are removed in the script as correlations at this locations do not reflect changes on the glacier. They were assigned a grid location of 1 to serve as a control for the other samples. Total number of grid points from the DIC algorithm was 9401.

Batch Five (DIC Grid to real-world sample points)

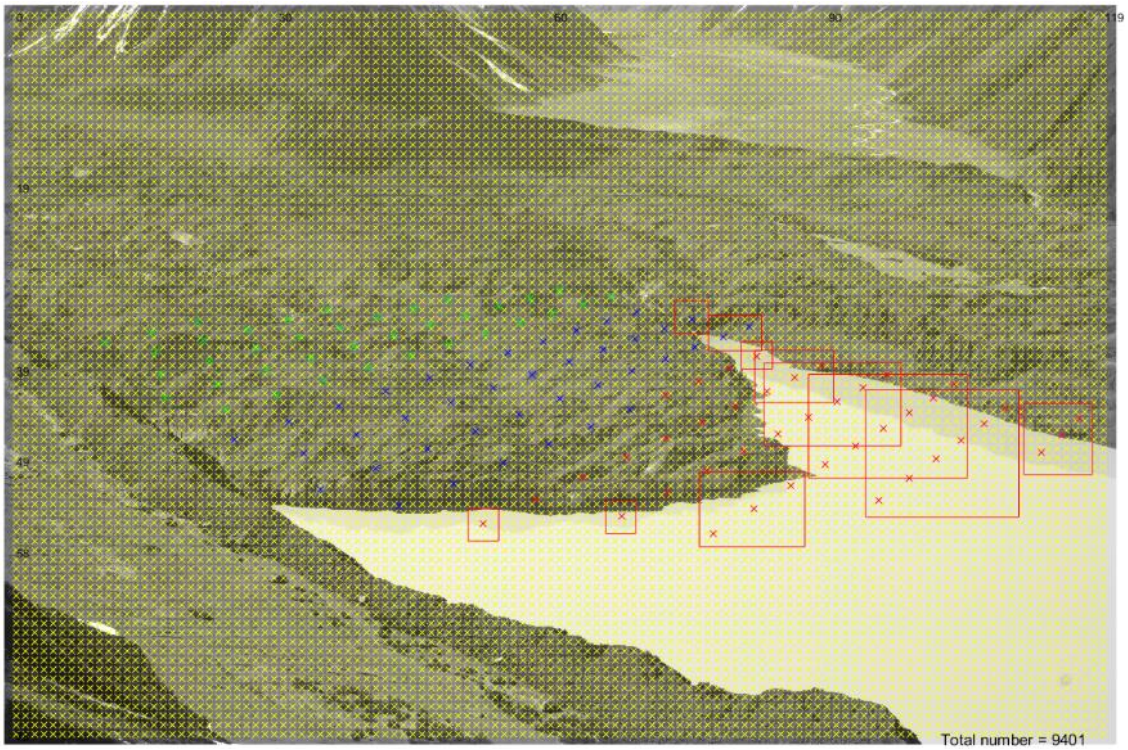


Figure Appendix 2.5. Batch Five data translation. Highlighted sample points are removed in the script as correlations at this locations do not reflect changes on the glacier. They were assigned a grid location of 1 to serve as a control for the other samples. Total number of grid points from the DIC algorithm was 9401.