Glacier Image Velocimetry: an open-source toolbox for easy and rapid calculation of high-resolution glacier-velocity fields

Maximillian Van Wyk de Vries[1,2] and Andrew D. Wickert[1,2]

[1]Department of Earth & Environmental Sciences, University of Minnesota, Minneapolis, MN
[2]Saint Anthony Falls Laboratory, University of Minnesota, Minneapolis, MN

Correspondence: Maximillian Van Wyk de Vries (vanwy048@umn.edu)

Final response - Review 2

Editor comments are given in *italic*, responses in regular font.

The toolbox presented by the authors is a GUI to ease feature tracking routines within a MATLAB environment. The naming of the toolbox is a reference to Particle Image Velocimetry (PIV), which is a well established technique within fluid mechanics. It is based upon particle seeding within a fluid, illuminated by light behind a narrow slit to get a within-plane velocity profile, thus in a well-controlled environment. When particles are not well stirred, have a lot of out-of-plane displacements, too large time separation, or other effects complicating the tracking: the experiment is done all over. The authors lend the methodologies from this domain, and implement this on natural environment to extract glacier surface velocity. This transfer is more complicated as it might initially seem to be, as seeding is absent (though people in hydrology are experimenting on this issue [Pizarro et al. 2020]), and more importantly experiments can't be redone.

The toolbox has some application specific adjustments, which is in line with other application domains, where similar toolbox are introduced, such as: Optical tracking velocimetry (OTV) [Tauro et al 2018], Surface Structure Image Velocimetry (SSIV) [Leitaoa et al. 2018], Kanade–Lucas Tomasi Image Velocimetry (KLT-IV) [Perks 2020], Part2Track [Janke et al. 2020], TecPIV [Boutelier 2016], Rectification of Image Velocity Results (RIVeR) [Patalano et al. 2017], Waves Acquisition Stereo System (WASS) [Bergamasco et al. 2017], or more specific to glaciers: Environmental Motion Tracking (EMT) [Schwalbe et al. 2017], PyTrx [How et al. 2020], Image GeoRectification and feature tracking toolbox (IMGRAFT) [Messerli et al. 2015], Pointcatcher [James et al. 2016]. Given the list above, and the journals where these toolboxes are presented, it might be a valid question why the authors have opted for a submission to The Crysophere, and not a technical EGU journal like Geoscientific Instruments or Geoscientific Model Development?

We thank the reviewer for their detailed review. We would specifically like to thank the reviewer for taking the time to read through GIV's code, as their recommendations have helped improve both the manuscript and the toolbox itself. We have taken the time to review each comment and recommendation in detail, and have made some moderate to substantial changes to GIV's code. We hope that this response clarifies some of our choices, and highlights the changes made.

The main modification which have been made to the code are:

1. We review the entire code to improve the code formatting (removal of unnecessary spaces and tabs) and increase the number of comments. We do not expect a large proportion of GIV's users to be reading through the entire code, but aim to make it accessible for when they do (and make it possible for users to use our functions in their own tools).

2. We change the format of our inputs array (which stores the input parameters) from a MATLAB cell array to a MATLAB struct array. The struct array uses dot notation to make it more human-readable, and should make the code more accessible (e.g. changed from 'inputs{30,2}' to 'inputs.parralelize').

3. We implement an option to read in images directly as geotiff images and preserve the geographic information throughout processing. If users input geo-tiffs they do not need to input corner coordinates.

4. We review the calculation of circular statistics (flow direction mean and standard deviation), and edit Dr. Berens' CircStat toolbox to tolerate NaN values.

5. We re-compile the Windows standalone app with the changes, and compile additional apps for macOS and Linux.

We describe the changes in more detail below, and respond to specific review comments. We have added reference to the relevant toolboxes you have listed in this paragraph as well.

To briefly touch on the decision to publish in The Cryosphere (TC) rather than a technical journal, the primary objective of our toolbox is to be accessible to glaciologists who may not otherwise be involved in feature tracking. We hope that GIV will bridge a gap between coarser resolution global glacier surface velocity datasets (e.g. GoLIVE, ITS_LIVE) and the more localised, higher resolution data required by some studies or field campaigns. Other good feature-tracking toolboxes do exist (see table 1 for many of them), but can be challenging to use with no computational or remote sensing background. GIV is easy to install, quick to learn, and quick to run, without requiring background knowledge of how the code runs (including by students). We believe that the manuscript is well within the scope of TC, and that publishing in TC will better reach the intended audience of this paper than technical journals. In addition, we believe that the case studies presented here will be of interest to the glaciological readership of TC.

We should note that we did share your concerns at the outset, and indeed contacted the editorial staff at The Cryosphere about appropriate fit before submission. Their support for our submission was also key to this being the chosen venue for the paper. This is often a difficult choice when presenting a paper focused on both the application and its methods, and we appreciate the acknowledgment of this.

There are some implementations which one could expect for a toolbox tailored towards glaciology to be presented, but are not included. In addition some features are included, which need further assessment, to highlight their improvement, if they do.

We thank the reviewer for highlighting some recommended changes. We describe the changes we make to the code in this

response.

To be more specific the comments are grouped into different paragraphs: Analysis of the 10-bit data of Landsat 8 [Jeong et al. 2015] and Sentinel-2 [Kääb et al. 2016] over dark overshadowed terrain have shown there is a significant benefit over 8bit data. Why do the authors downgrade their imagery to 8-bit RGB? In the same line, the use of non-georeferenced ".jpg" or ".png" data is strange. MATLAB supports mapping tools for such issues, why are these not adopted. While the toolbox needs a specific dimension in order to work, it is very strange this is not present in the toolbox. The choice of using angles (lat,lon) for metric data is confusing.

In brief, the changes we make to the code now make it possible to run georeferenced, 10 bit geotiff datasets. When applying an orientation filter to the data (e.g. the NAOF we have implemented), we find little difference between 8 bit and 10 bit data even in shadowed/clouded areas. The orientation filter tends to cancel out contrasts within the data. The 10 bit data may prove advantageous where orientation filtering is not suitable and simple low-pass filtering and/or contrast limited histogram equalization (CLAHE) is used.

The authors present a new image transform dubbed "NAOF", but is there a significant improvement, over for example typical orientation filtering approach? At least to me, the 45 degrees filters seem like a redundant feature, as 90deg angled steerable filters incorporate all information [Freeman et al. 1991]. It seems the implementation is a combination of work similar to [Ahn et al. 2011], and orientation filtering [Heid et al. 2012]. But it is questionable, if the additional calculations add towards improvement. The filter banks are correlated (as just mentioned), furthermore, the visible bands are similiarly correlated over glaciers. Hence, the information gain might be very limited.

We tested a number of different kernels for the orientation filter, and found that some of the simpler orientation filters would enhance and suppress features (particularly crevasse fields) depending on their orientation. The filter banks are correlated, but adding both the 90 degree and 45 degree filters preserves more feature uniqueness than either alone. This is important for reducing the number of false matches and signal to noise ration in crevasse fields, where subsequent crevasses have similar orientation signals. The visible bands are correlated, which is why we sum them into a single band. On some glaciers we have found that near infrared Band 8 of Sentinel 2 (842 nm) may produce a better feature contrast than other bands on some glaciers. We have also found that shortwave infrared bands 10 and 11 (1375 and 1610 nm) can be suitable in some cases (high contrast between ice and supraglacial debris), however suffer from a lower spatial resolution. Different band combinations can easily be created within SentinelHub (which we discuss briefly in the user manual). We mostly use contrast-enhancing SentinelHub custom script (javascript) which combines Sentinel 2 bands 3,4 and 8. We will add this custom script to a GitHub repository.

The paragraph on velocity calculations talks about two methods to extract displacements from an image pair. However, this distinction is more about the option if refinement is applied or not. The authors limit themselves to frequency domain methods,

which is the de-facto procedure for PIV. However, spatial domain methods can be more favorable for glacier related applications. This comes down to the point given above, of the transfer from fluid mechanics towards an application domain without the ability to redo an experiment. Hence, the velocity profile can be highly variable (extensive shear, valley walls, diverging flow, fast flow, ...). Spatial domain methods are more resistant to shear flow, the chip is not related to the search space (as with fourier methods). The authors might have considered spatial domain methods, but for clarity it might be fruitfull to mention this, and why. In PIV experiments, one is able to adjust the image sampling rate, to be in accordance with the maximum flow velocity, this is less the case for glaciers.

At the early stages of this toolbox we investigated both frequency domain and spatial domain (Normalized Cross Correlation) matching algorithms. In particular we tested the NCC option within IMGRAFT (Messerli and Grinsted, 2015) and 'discrete cross correlation window deformation' option in PivLab (Thielicke and Stamhuis, 2014) which allows for shearing of features. We found no clear improvement in matches relative to frequency domain cross correlation even in areas with significant ice shear (glacier margins with around 1000 m/yr velocity gradient across one kilometre). We also found that the multi-pass frequency domain matching would usually outperform single passes in both frequency and spatial domain. However, we found that changes to the pre and post-processing would often have the greatest improvement on final velocity maps, so have emphasized this aspect in GIV. This is indeed less of an issue within fluid dynamics where the sampling rate, tracers and illumination may be controlled.

We have added two sentences explaining why we opt for frequency domain methods in GIV, and some of the relative benefits of each method. We also refer readers interested in further discussion to Thielicke and Stamhuis (2014) and a chapter of a recent PhD thesis by Altena (2018) which provides an accessible and detailed discussion of these topics from the perspective of glaciology.

Technical comments: In general the code is very messy, many duplicate lines exist, commenting is absent ("strcmpi(inputs{51,2}, 'Yes')"). At multiple locations indents are used in-appropriately. Documentation and commenting within the code is limited. Every function has an extensive help section, which is typically an acknowledgment and info about GIV. Input or output variables are not described. For many or all functions, tests are lacking. Factorization is on a low level. Given this situation of the code, how do the authors see adoption by others? The objective of the authors is to be a wrapper (based upon ImGraft and matPIV), then one would expect documentation to be extensive.

We would again like to thank the reviewer their time taken providing feedback on our code. We have inspected every function in GIV to improve formatting (tab only within loops, no unnecessary spaces, etc.), increase the number of in-code comments, and add to function descriptions. As mentioned above, reading the code is not necessary for running GIV (and is not possible in the case of the standalone apps). As such we do not believe code commenting is an important factor for the adoption of GIV by others. However, we aim for our code to be scrutable by those that do wish to read it, or wish to use our filters within their own code.

A number of the 'duplicate lines' within the code are present in order to enable the various input options. We hope that the additional comments throughout the code, alongside the change of our inputs array to a dot notation struct array will make these clearer. For example "strcmpi(inputs{30,2}, 'Yes')" should now read "strcmpi(inputs.parralelize, 'Yes')" and be accompanied with at least a brief comment.

Sub-pixel: There does not seem to be a sub-pixel estimation present? Why is this?

A sub-pixel estimation is present. For the single-pass algorithm it is built into the 'GIVtrack.m' function, while for the multi-pass algorithm a separate function ('GIVtrackmultipeak.m') is called within the final feature tracking pass ('GIVtrackmulti-final.m'). The multi-pass (recommended) sub-peak finder is set to a gaussian peak fit by default (and in the app), but can be modified to a centroid or parabolic fit within the code if desired. We have found that changing this has little effect on outputs.

The domain of PIV is also extensively populated with papers about peak-locking, it would be good if the authors put some effort in this as well. This bias is especially present in sub-pixel estimation directly done on the correlation surface. Other methods, such as TPSS, as implemented in Cosi-Corr are less sensitive to this effect (which to a large extent might explain its popularity in geodetic imaging).

We have not investigated peak-locking in detail while working on GIV. Inspection of raw displacement and velocity histograms shows only a minor bias towards integer values, and does not appear to be a major source of error.

A project is currently underway comparing feature tracking results from GIV and other feature tracking toolboxes to GPS derived velocity data. We are planning on considering the effect of different sub-pixel estimation schemes, and future versions of GIV will be updated if a particular algorithm is clearly superior. An in depth discussion of the particle tracking parameter options is beyond the scope of this manuscript.

Geo-referencing: The gridspacing as it is implemented now assumes all rectangular and upright pixels, please adjust this.

As mentioned above, we have added an option for GIV to process raw geotiff data and read in geographic data off the images.

If this comment is referring to the velocity fields being projected onto a horizontal plane (i.e. not corrected for topographic strike and dip), this is common practice in feature tracking. This correction may be applied using a DEM after running GIV, but is usually very small.

Matlab: To my knowledge Matlab is not open access, the code might be open, but many of the algorithms are hidden away and licensing is needed. Hence, I am not so sure if the description about FFTW is needed, as one is not able to access this. Secondly, it is a standard routine within MATLAB.

FFTW is an open source toolbox, however this is correct about much of MATLAB. The objective of this paragraph is to provide some justification for the toolbox being coded in high-level interpreted programming language MATLAB. FFTs constitute the largest computational expense of feature-tracking models, and in MATLAB are performed efficiently in C subroutine library FFTW rather than in any native MATLAB code. As such, GIV can process feature tracking pairs rapidly (particularly with the built-in parallelisation).

Merging satellite data: The authors describe a time-series construction method, illustrate a toy example. But it seems they apply a sigma-filter, and do simple infilling. There are more advanced methods available, which are more adaptive towards velocity data. Hence, a simple reference to (e.g.: [Mouginot et al. 2017]) might suffice for the implementation here, and this section can thus be reduced considerably.

The temporal sampling is a bit counter intuitive, as this is an opposite direction to the workflow of [Millan et al. 2019]. Where they found the time-span needs to be of sufficient size, in order to be of most use. Why then do the authors prefer the shorter time steps?

We do not fully understand the above comments. Our findings on optimal temporal sampling are in line with Millan et al., 2019, and we by default exclude velocity image pairs with temporal separation of less than one week. We describe the example of a very slow moving glacier in the Chimborazo case study, where excluding time steps shorter than 6 months provided the best velocity results. The minimum and maximum time separation limits can be adjusted within the user interface according to the specific characteristics of the glacier of interest.

If this comment is referring to the first iteration of monthly timeseries generation, the 0-1 scoring system refers to the proportion of a velocity map within a given month rather than the temporal separation. A 7 day or 30 day separation velocity map entirely within a given month will both be assigned a score of 1. Velocity maps overlapping into other months will be assigned lower scores.

The sigma-filter and infilling are post-processing steps to improve the quality of individual monthly maps.

Other averaging and timeseries generation methods for glacier velocities are available (e.g. Millan et al., 2019's NetCDF Geo-Cubes, Altena et al., 2019's Hough space method, etc.), although do not inherently generate monthly time-series. Our method generates a full dataset mean and median, as well as monthly velocity maps which (we hope) are easier to interpret than the unevenly sampled satellite image pair timing.

Code specific: There is a function about intensity capping, while this might be of interest to PIV, it is questionable if this is of interest to environmental signals. Bright intensity of seed points within dark water are very much different from glacier surfaces. Also how does this merge with the high-pass filtering... ?

Intensity capping is not usually necessary, but can be useful where bright snowpatches are present close to a debris covered or otherwise dark glacier. In general the filters do not need to all be applied or stacked. By default only the orientation filter is applied. My recommended filter choices are:

- Orientation filter (NAOF) only = default
- High-pass filter and CLAHE (+ Sobel filter in some cases) = where orientation filtering creates too much noise
- Intensity cap + High-pass filter and CLAHE (+ Sobel filter in some cases) = where bright patches are present, and orientation filtering creates too much noise

Users may wish to experiment with a small dataset, different glaciers will have different optimal pre-processing parameters.

Even though your algorithms are optimized, they might not pull out the best of the machinery, yet. For example, the function "neighbourfilter" contains a double loop, to process a kernel. An internal function of Matlab called "nlfilter" might be much faster. A good reference for such implementations can be found in "Accelerating MATLAB Performance" (http://undocumentedmatlab.com/books/matlab-performance). In the same function from line 95 onward, the authors use double indexing. While linear indexing is possible as well, which make vectorization possible (which is MATLABs strong point). This greatly improves the processing time as well.

We thank the reviewer for the useful reference and recommendations. We test an implementation of 'neighbourfilter.m' using nlfilter, however it results in little overall speed-up and some unexpected artefacts. The artefacts can probably be fixed by re-writing the function around nlfilter, however due to the small overall increase in efficiency we leave the function as is for now. We believe that the inclusion of parallel computing of feature-tracking pairs is likely to result in the greatest computational speed-up relative to other tools.

I am not sure if the variance calculation of the flow direction is calculated correctly. Offsetting the direction and applying a weighting might work, but maybe correct circular statistics might be more appropriate, see [Berens 2009] for a toolbox implementation. It might be more easy to use the mapping coordinates instead...?

Our flow direction mean calculation appears give the same results as Berens's CircStat on examples tested, but are not sure of its behaviour in all scenarios. Thus, we replace it with the well-tested CircStat.

We make several changes to CircStat to enable calculation of mean and standard deviation in data containing Not a Number (NaN) values. We will share the NaN tolerating version of the toolbox on GitHub or file exchange.

It might be good to give an example; here is a sniplet of your code (a straight copy):
if smoothsize == 2;

mask = [0 1 0; 1 0 1; 0 1 0];
elseif smoothsize == 3;
mask = [0 1 0;1 1 1; 1 0 1; 1 1 1; 0 1 0];
elseif smoothsize == 4;
mask = [0 0 1 0 0 ;0 1 1 1 0 ;0 1 1 1 0; 1 1 1 1 1 ; 0 1 1 1 0; 0 1 1 1 0; 0 0 1 0 0];
elseif smoothsize == 5;
mask = [0 0 1 0 0 ;0 1 1 1 0 ;0 1 1 1 0;1 1 1 1 1 ;1 1 1 1 1 ; 1 1 1 1 1 ; 0 1 1 1 0; 0 1 1 1 0; 0 0 1 0 0];
end

why isn't this in a function? Something like: function [mask] = make_mask(smoothsize)
"
generates a neighborhood kernel in the form of a diamond,
though excludes the central pixel
input:
smoothsize - integer value
output:
mask - logical array
"
if nargin<1, smoothsize = 3; end
mask_radius = floor(smoothsize/2);
mask = strel('diamond', mask_radius); % make a diamond shape
mask = mask.Neighborhood;
mask(mask_radius+1, mask_radius+1) = 0; % exclude the central element
end

That function is indeed much more elegant and flexible than our implementation. We have implemented it in the code (with due acknowledgement) and hopefully it makes the filter more flexible for other uses.

In all I am aware this is, like all research, work in progress, and I think this is a very useful direction. But in its current state and form, sufficient work needs to be done to be of interest, please see [Perks 2020] for a nice example. Here workflows are nicely presented, a dashboard is present, etc. All in all, I think a transfer towards a technical journal of EGU might be more in place.

We hope that the comments and code edits presented clarify our decision to submit to The Cryosphere, and answer some of the concerns presented in this review.

Textual: p63: "broken down" is misleading, as overlapping chips can also be used p140: superscript the 2, to make it a square

The recommended changes have been made.