Glacier Movement Prediction through Computer Vision and Satellite Imagery

Maria-Minerva Vonica
Faculty of Mathematics and Computer Science
West University of Timişoara, Romania
Email: maria.vonica98@e-uvt.ro

Andrei Ancuta
Laboratory of Geo-information
Science and Remote Sensing
Wageningen University, the Netherlands
Email: andrei.ancuta@wur.nl

Marc Frincu
School of Science and Technology
Computer Science Department
Nottingham Trent University, UK
Email: marc.frincu@ntu.ac.uk

***Abstract*—Over the last decade, climate change has impacted Earths' atmosphere and environment more than anytime before. Glaciers are the most sensitive indicators of its impact. In this work, we model a glacier's evolution by applying computer vision algorithms on high-resolution satellite imagery. We detect changes in the ice coverage movement by applying a dense optical flow algorithm over an image time series covering a particular scene (region) and processed to extract the NDSI. We perform tests on the Jungfrau-Aletsch-Bietschhorn (JAB) glacier in the Swiss Alps. Our results show that we are able to obtain relevant information by computing motion vectors across time. Furthermore, we observe small differences between our predicted NDSI and the observed values demonstrating the efficiency of the approach. In addition, by applying various machine learning techniques we provide an optimistic date for when the JAB glacier will completely disappear: 2800.**

***Index Terms*—Glacier retreat, Computer vision, Satellite images, Landsat 8, Image alignment, Dense Optical Flow, Jungfrau-Aletsch-Bietschhorn glacier**

I. Introduction

According to a report delivered by NASA in March, 2021, Earth's average surface temperature which was recorded in 2020 came to be equal with the temperatures which designated 2016 as the hottest year on record [1] and they are expecting that records will continue to be broken.

Analysing temperature trends on a global scale provides vital indicators such as carbon dioxide levels in the atmosphere, which are now growing more than they can be naturally eliminated, causing an increase in phenomena such as sea ice and ice sheet mass loss, heat waves which are will become more intense and longer, rising level of the sea, as well as changes in the fauna and flora of the planet.

Having a mature enough approach for timely identification of these climate trends represents an essential necessity for human life. Changes in environment will require changes of approach to problems such as managing water resources, creating different crops which can withstand extreme changes of temperature and being prepared for potentially disastrous weather events.

Since glaciers are known the be the most sensitive indicators to climate change, extraction of information about their changes in the last years could prove valuable. Glacier retreat is happening due to increased values in temperature causing less snow fall and less accumulation of ice in time.

For instance, the largest glacier in the Alps, Jungfrau-Aletsch-Bietschhorn, retreated by about 10.5 hectares between 1985 and 2019 as it can be seen in Figs. 1 and 2. Studies performed in 2011 have indicated that by 2100 almost 90% of its ice volume will disappear [2].

Fig. 1. Change in the Jungfrau-Aletsch-Bietschhorn glacier: 1985 (left) and 2020 (right).

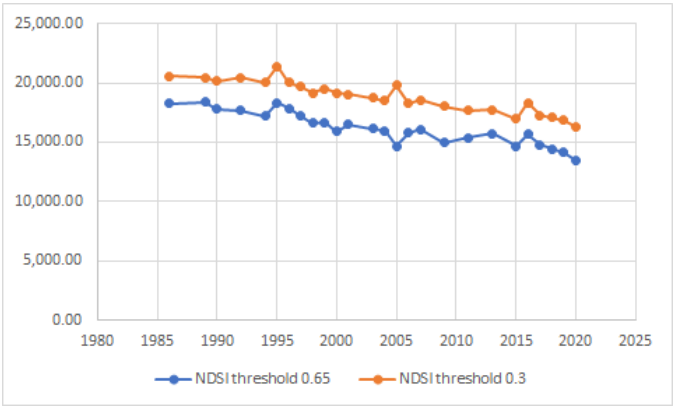

Fig. 2. Evolution of the Jungfrau-Aletsch-Bietschhorn glacier between 1985 and 2020. The y-axis represents the number of snow pixels extracted by applying NDSI (Normalized Difference Snow Index) with 2 threshold levels.

The goal of this paper is to model glaciers by analysing the optical flow between consecutive satellite images. Based on information extracted from datasets constructed over the

past decade, we are creating predicted images which could highlight future glacier retreat and different patterns of accumulation. Since current methods of large dataset acquisition do not scale, we are also creating a tool for easier search and download.

The rest of the paper is structured as follows: Sect. II provides an overview of the related work; Sect. III describes in detail the proposed solution; Sect. IV summarizes the experiments and their results; and finally Sect. V outlines the main conclusion and future work.

II. State of the Art

In [3], the authors use both Sentinel and Landsat satellites for image collection in order to produce a higher quantity of qualitative images for glacier mapping, while also covering wider areas. They propose four approaches for studying automatic glacier mapping. The first analyzes different band ratios for multispectral image creation. The second proposes robust methods used for improving glacier mapping by exploiting the seasonal variation yielded by the spectral properties of snow. The third highlights spatio-temporal variation of glacier surface types. Finally, the fourth analyses how the chosen band ratio images generated from the first application can be used for automatically detecting changes in glaciers. Their results show that the alpine and arctic advances and retreats provide the most visible signs of climate change and they propose to revisit current methods for glacier mapping in order to reduce the practice of manual glacier mapping.

Glacier sensitivity to climate change is also addressed in [4]. The authors propose a remote sensing based framework used on multispectral satellite imagery for studying the ablation and accumulation processes of glaciers in order to quantify their mass balance. They study the mass loss between 1998 and 2016 for the Parvati glacier, located in the western Himalayas. Their results show that the glacier responds to climate change factors each year through a high mass loss, resulting in a strong effect on water availability and river flows.

In [5], the authors propose a decision-based image classification algorithm for separating ice, debris and snow surfaces on glaciers and extracting snow lines both monthly and annually. They achieved this by automatically partitioning glacier surfaces through band ratios combined with topographic criteria which are extracted for each pixel. They have conducted the study on the Karakoram and Trishuli regions located in eastern Himalaya, with images taken between 2000 and 2016. They have concluded that snow lines are the most sensitive to manual corrections, elevation dataset, topographic , and the calculated thresholds for the band ratios for the spring and winter months.

In [6], the authors propose descriptive methods for glacier mapping by using aerial time series produced by the Synthetic Aperture Radar sensor from Radarsat-2 and Sentinel-1A. Out of their five scenarios, the first tracks transient snow lines and it proved to correlate with both Landsat 8 and Sentinel-2A produced aerial images. As a conclusion, they state that automatically derived satellite imagery products prove to be important in analysing glacier change.

Another interesting approach to determine glacier melt was analysed in [7] where the authors have compared the band ratio method with the Normalized Difference Snow Index (NDSI) one for extracting parameters which highlight glaciers. They have conducted their studies on the Karakoram area with satellite imagery collected from Landsat 8 starting from 2014. Their results show that for boundary extraction, the band ratio technique yielded better results. They also state that visual interpretation is still an major factor in analysing the obtained results.

In this paper we rely on another idea which tries to capture movement in consecutive images. Farnebäck, Gunnar proposes an interesting approach in [8] in which they present an algorithm for estimating flow in images by approximating each neighbourhood through quadratic polynomials. Their transformation under translation is then analysed in order to extract displacement fields. This method of computer vision has not been tried yet in the field of glacial analysis and it has potential to yield interesting results.

III. Proposed Solution

Based on the already existing approaches as described in Section II, we propose to extract the snow and ice sections of the images by using the normalised snow difference index, as explained in [7] and analyse movement of ice by calculating the dense optical flow between pairs of time consecutive satellite images, based on the algorithm created by Farnebäck, Gunnar, as described in [8].

The processing workflow can be seen in Figure 3. The next sections will describe each step in greater detail focusing on the technical challenges and algorithmic solutions.

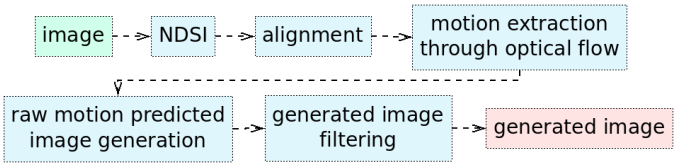

Fig. 3. Processing diagram.

A. Image collection and NDSI processing

We have chosen to work with images collected by the Landsat 8 satellite mainly because their archive is structured such that the images are consistent over time, of high quality, and fit for time series analysis. On top of this, since the satellite has a cycle of 16 days, there are over two million freely available images, dating from 2013 until September 2021.

One of the most popular services for satellite imagery downloading is USGS Earth Explorer. It can query and order aerial data collected from various satellites. The tool is particularly useful when the main focus is to analyse a specific area rather than trying to acquire a large dataset of scenes. However, downloading a large set of images proves to be rather difficult as the parameters for each scene need to be manually set. On top of this, the query results have to be manually picked and then passed for downloading through another application which handles their bulk download.

In order to fix the problem of excessive manual labour which appeared by using the USGS Earth Explorer we implemented an endpoint of the *SpatioTemporal Asset Catalog API*[1]. A script is used to bulk search and download based on an already existing database of over 130,000 entries representing glaciers and their afferent parameters (e.g., location, elevation, name).

By using this method we can choose which glaciers we want to download directly from the database and the dataset will be automatically structured as they are fetched. Images containing clouds can be dropped by setting a maximum allowed cloud coverage. This tool works as a plug-in and it is independently executed from the main application.

Next we must extract ice and snow pixels from the downloaded images. For this we have decided to use the **Normalized Snow Difference Index** (NDSI) measure with a threshold of 0.3, which is obtained from Eq. 3.

$$NDSI = \frac{green - swir1}{green + swir1} \tag{1}$$

Snow and ice usually have a very low reflectance in the shortwave infrared spectrum and very high in the visible one, which is useful for removing most types of clouds from the scene [9]. We can then apply the thresholding function from Eq. 2 to separate snow free land from covered one. Different threshold levels produce different snow covered pixel areas (cf. Fig. 2). In this paper we use a value of 0.3.

$$thresholding(pixel) = \begin{Bmatrix} \text{snow/ice}, & NDSI \geq 0.3 \\ \text{snow-free}, & NDSI < 0.3 \end{Bmatrix} \tag{2}$$

To aid the visualisation we have coloured the result such that snow pixels are white, while snow free land is green. The resulting NDSI image can be seen in Figure 4 for the Jungfrau-Aletsch-Bietschhorn glacier in the Swiss Alps.

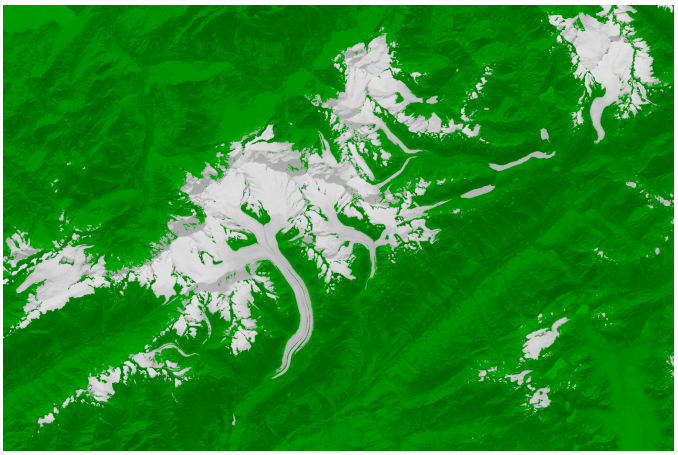

Fig. 4. NDSI image of scene A: Jungfrau-Aletsch-Bietschhorn glacier.

[1]http://nsidc.org/data/glacier_inventory/index.html

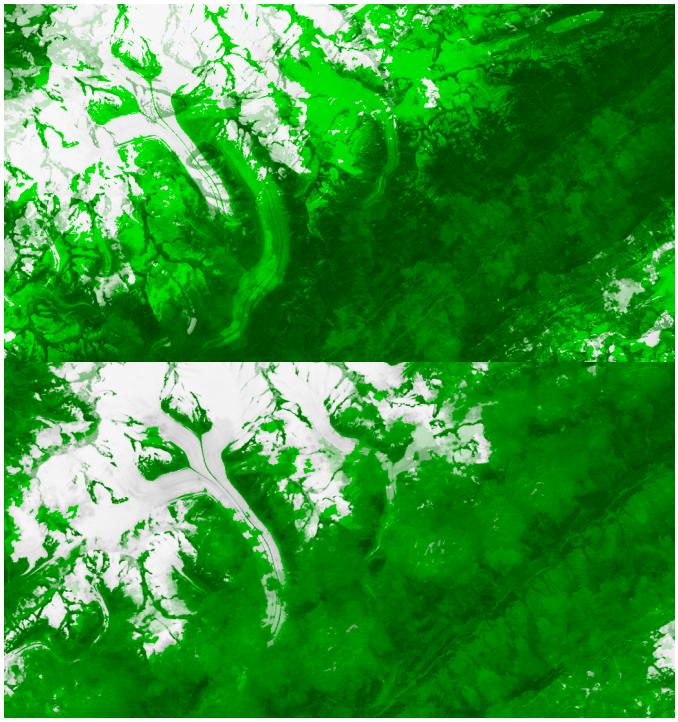

Fig. 5. Overlapped unaligned (top) and aligned (bottom) NDSI scenes B and C.

B. Alignment

Landsat's trajectory orbiting Earth is not precise, therefore not all images will overlap at pixel level. The *green* and *swir1* bands have a spacial resolution of 30 m, therefore a misalignment of just 50 pixels between two images can lead to a 1.5 km offset. Tracking pixel motions through a series of images without aligning them first would yield in erroneous results, due to inconsistent geographical coordinates. Figure 5 highlights the misalignment between scenes B and C.

We have solved this problem by using a strong keypoint detection algorithm which collects features from all bands within an image and matches them with a given reference. In our case we found that the most reliable keypoints were around mountain edges and other geographical features present in the image. We have used a combination of *ORB (Oriented FAST and Rotated BRIEF)*, *Harris Corner Detector*, and *RANSAC (Random Sample Consensus)* which we applied on the raw 16bit grayscale bands of each scene.

ORB represents a fusion between the features extracted by using the fast accelerated segment test (FAST) keypoint detector combined with the binary robust independent elementary features (BRIEF) descriptor. The FAST detector finds keypoints in the image and uses Harris corner measure to select a number of top points from the generated list. We have found that keeping the top 25% matches extracted from 5,000 keypoints yielded the best results. Splitting the image into multiple boxes and applying the ORB algorithm on each separate part of the image such that features are homogeneously distributed also improved valid match making. As a final optimisation we set the maximum allowed shifting

euclidean distance between any two pixels to be at most 200 (6 km on the map) therefore getting rid of outlier matches based on the distance.

The obtained matches can be then used to create an affine transformation matrix, generated through the RANSAC algorithm, proposed by Fischler and Bolles [10]. The affine transformation matrix describes the rotation and translation of the image to be aligned in comparison to its reference and it is used for warping it such that no two pixels are misaligned.

C. Movement extraction through optical flow

From our series of satellite images we can **track the motion** of each pixel from one image to the next. This effectively allows us to generate new images from information extracted from previous ones.

Extracting the motion vectors between two consecutive images can be achieved by calculating their **optical flow**. Optical flow is defined as the motion of objects between consecutive frames of a series, produced by the relative movement between the object and camera. By using computer vision algorithms which calculate the optical flow of two scenes, we can track the changes in glacier retreat across a time series dataset by estimating their current velocity and predicting their position in the next images.

Figure 6 depicts this idea, where we can express an image as a function $I : N^3 \rightarrow N^3$ of 2D space with coordinates (x, y), and time t. If we take the first image to be $I(x, y, t)$ and we move its pixels by a distance of $\Delta x, \Delta y$ over a timeperiod Δt we obtain the new image as $I(x + \Delta x, y + \Delta y, t + \Delta t)$.

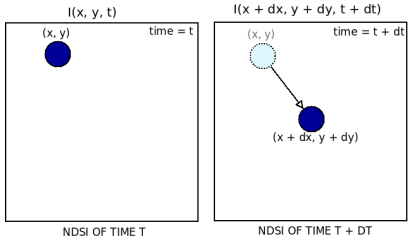

Fig. 6. Optical flow between $NDSI(time = t)$ and $NDSI(time = t + \Delta t)$.

There are multiple types of optical flow generation algorithms, but for the purpose of our paper, we have chosen the **dense optical flow** algorithm proposed by Farnebäck, Gunnar [8]. Even if dense implementations have a higher cost we chose to make this trade because it calculates the motion for each pixel of the frame while also having a higher accuracy compared to sparse methods such as in the one proposed in [11].

The best results have been generated with a pyramid scale of 0.5 and 6 pyramid levels keeping in mind that the images that we work with are large. At each level, the image is going to be reshaped at half the size of the previous one; therefore the search area for motion is small enough for optical flow to track movement, with 3 iterations over each layer. The rest of the parameters have been been set at their typical values.

The results contain the computed motions for each pixel, treated as a pair representing the distance and direction that its coordinates moved from one image to the next (cf. Figure 6). As an example, the generated optical flow vectors for scene D can be seen in Figure 7. To avoid visually flooding the image with vectors we are only drawing them every 30 pixels (900 m).

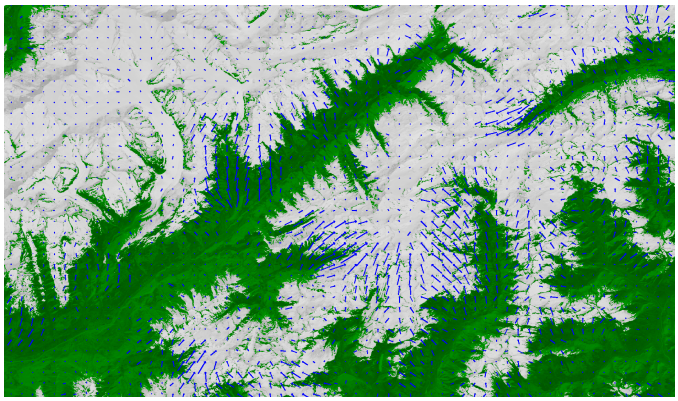

Fig. 7. Overlayed motion vectors generated by optical flow for image D.

D. Generating predicted images based on motion

Optical flow acan detect the movement of the ice front. Therefore we propose to generate NDSI images whose pixels will be predicted based on the past extracted motions for a given frame. We can obtain the motion predicted NDSI image by relocating each pixel value from the $NDSI(time = t)$ to a new location $(x+2\cdot\Delta x, y+2\cdot\Delta y)$, effectively generating the new image $NDSI(time = t + \Delta t)$ as described in Figure 8: $(I(x + 2 \cdot \Delta, y + 2 \cdot \Delta x, t + 2 \cdot \Delta y)$.

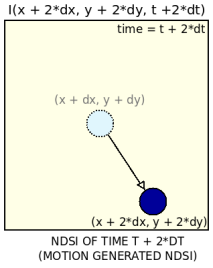

Fig. 8. Overlayed motion vectors generated by optical flow for scene D.

As a first approach of populating the new image, we have simply iterated over all the pixels in $NDSI(time = t + \Delta t)$ image and calculated their new coordinates based on their respective motion vector (cf. Algorithm 1). Since this is an iterative approach, it does not scale for images as large as the ones we are using. Furthermore, since we are using the Python language, which is an interpreted one, each operation has to go through an interpreter before being processed. For very granular data processing this proves to be inefficient. For a typical image of 8543×8039 pixesl it took as much as 10 minutes for generating the image on the machine that we have used for processing (cf. Section IV-G).

Algorithm 1: Algorithm used for motion predicted image generation based on the optical flow vectors and $NDSI(time = t + \Delta t)$

```
1  function generate (NDSI, motion_vectors);
   Input  : NDSI(time = t + Δt and the motion
            vectors generated by optical flow between
            NDSI(time = t) and
            NDSI(time = t + Δt)
   Output: The predicted image
            NDSI(time = t + 2 · Δt)
2  motion_predicted_image ← NDSI.shape;
3  width ← NDSI.width();
4  height ← NDSI.height();
5  for y in [height, width] do
6      for x in [height, width] do
7          dx ← motion_vectors[y][x][0];
8          dy ← motion_vectors[y][x][1];
9          new_x ← x + Δx;
10         new_y ← y + Δy;
11         motion_predicted_image[new_y][new_x] ←
             NDSI[y][x];
12     end
13 end
14 return motion_predicted_image;
```

Since the data used for this paper is large in size, relying on Python alone does not scale. Python is an interpreted language meaning that for each operation its interpreter has to do extra work such that it translated the bytecode instruction into a form that is machine executable. To overcome this limitation we resorted to using **NumPy**, an open-source library which brings support of large data computation, such as multi-dimensional arrays and matrices, as well as a large collection of mathematical functions which can be easily used for their operation [12]. The library has a **well optimised C code core** which can handle large processing by bypassing the Python interpreter. Therefore, based on the needs of our dataset, we have used NumPys' main data structure, the ndarray (n-dimensional array) as a data structure to hold our images. Various optimisations applied throughout processing focused on using only this type of data structure in order to make use of NumPys' optimisations and speed. By doing this instead of simply iterating, we were able to improve the processing time by around 1,700%, reducing it to 35 seconds.

In the iterative approach, for each pixel we add its motion to its position such that we get its new location. These numbers can be computed ahead of time and transformed into arrays such that we only use optimised operations. By creating an array of the initial coordinates (x, y) and adding the motion vectors (dx, dy) array to it, we generated the absolute coordinates where each pixel from the NDSI image should be translated. By adding this modification we got rid of lines 5–11 from the algorithm and replaced them with the code from Algorithm 2. The $absolute\_coordinates$ and $NDSI$ can be treated as a sparse array which is then densified using NumPy capabilities.

Algorithm 2: Improved algorithm used for motion predicted image generation based on the optical flow vectors and $NDSI(time = t + dt)$

```
1  function generate (NDSI, motion_vectors);
   Input  : NDSI(time = t + Δt and the motion
            vectors generated by optical flow between
            NDSI(time = t) and
            NDSI(time = t + Δt)
   Output: The predicted image
            NDSI(time = t + 2 · Δt)
2  motion_predicted_image ← NDSI.shape;
3  width ← NDSI.width();
4  height ← NDSI.height();
5  index_array ← [[[0, 0], [1, 0]...[width, 0]
     [0, 1], [1, 1]...[width, 1] ...
     [0, height], [1, height]...[width, height]]];
6  absolute_coordinates ←
     motion_vectors + index_array;
7  motion_predicted_image[absolute_coordinates] ←
     NDSI;
8  return motion_predicted_image;
```

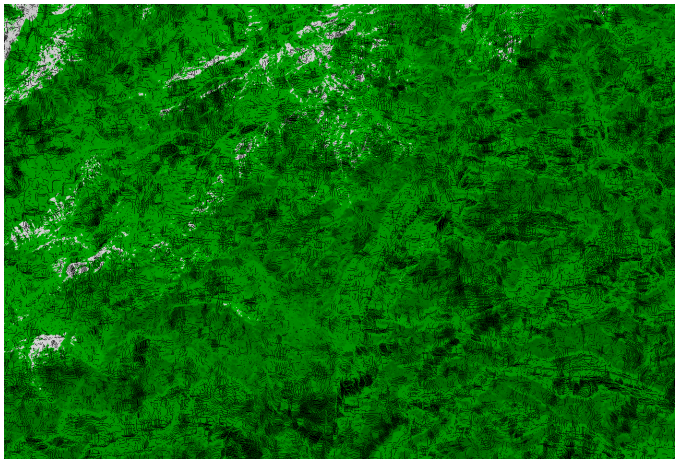

Fig. 9. Raw generated NDSI image, unfiltered, zoomed in for a clear view.

The function used to generate the predicted NDSI is non-surjective leaving some pixels undefined. This results in a noisy generated image as it can be seen in Figure 9 (black pixels). We solved this through a **filter** which uses the weighted average of the neighbouring pixels values to fill the missing ones.

E. Generated image filtering

A first step into creating the filter for the motion predicted NDSI image is to create a mask which will be applied on the black border of the scene such that we are looking for undefined values only inside it. By using NumPy, the **coordinates** of these pixels are extracted into an array which then can be processed in parallel by Pythons' multiprocessing library.

We have used multiprocessing instead of threading because Python does not have true parallelization in multithreading, only concurrency, which would not be a real improvement on the processing time. As all the processes have to write different chunks of the same image and they do not share the same memory space. We chose to solve this problem by creating a shared memory buffer to hold the image.

The value of each found undefined pixel has to be calculated as an weighted average composed of its neighbouring pixels, as shown in Figure 10 (top). We created a **kernel** which holds the weight of each pixel in the neighbourhood such that pixels closer to the centre have a higher weights than those near the edge. However, there are cases when multiple undefined pixels exist in the same neighbourhood. Since we cannot take their values in consideration we do not include them in the weighted average by setting their weight to 0 as shown in Figure 10 (bottom left). The result of the weighted average will be then stored as the value of the currently focused undefined pixel as highlighted in Figure 10 (bottom right).

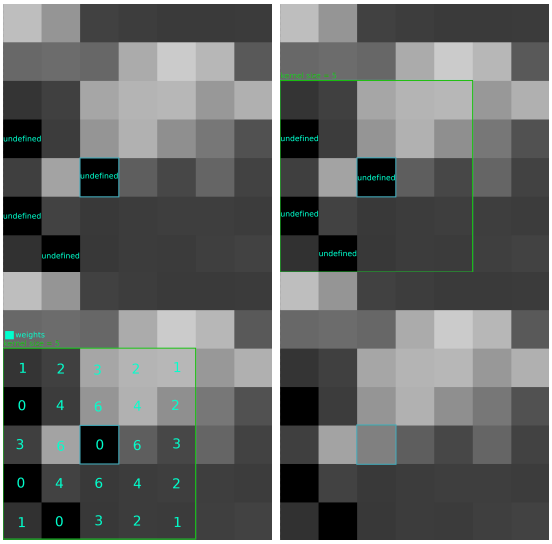

Fig. 10. Zoomed in part of the predicted NDSI (top left); extracted neighbourhood (top right); kernel of weights (bottom left); generated value (bottom right).

IV. Experiments

A. Dataset

For our tests we have used two well studied glaciers and downloaded all their public available satellite imagery with at most 20% cloud coverage. Due to different seasonal snow coverage fluctuations we have split our dataset based on $(path, row, month)$ pairs as suggested in [5]. The predicted image is overlapped onto the analysed one, and we have colourised them as follows: *green* represents the areas which are snow free, while *white* represents the areas which are covered, as described in Formula 2. The cyan areas represent areas which are predicted to develop snow buildup, and the orange areas are expected to melt. These two colours represent the error of motion prediction between the two images.

The first glacier region which will be analysed is named **Jungfrau-Aletsch-Bietschhorn**, located in the **Swiss Alps** $(46.477°N, 8.056°E)$ an elevation between 809 and 4274 m. This location is at the intersection of multiple WRS-2 coordinates, therefore we will three different sets of $(path, row, month)$ as depicted next. The images are captured between January 2013 and July 2021.

The second glacier which we have taken into consideration is named **Parvati** and it is located in **India**, $(31.754°N, 77.675°E)$ at a maximum elevation of 5599 meters. We have chosen this glacier in order to compare our results with those in [4]. They have chosen a dataset of images located at WRS-2 path 147, row 38. Unfortunately, we could not fetch the images for that specific path and row in order to be able to make this comparison. Instead our results are for path 146, row 38.

B. Evaluation metrics and approach

To asses the accuracy of our approach we have used the percentage of snow covered areas within an image as determined by applying NDSI. Furthemore, we have used MAPE (Mean Absolute Percentage Error) to estimate the difference between our predictions and the observed NSDI percentage values for each scene.

$$MAPE = \frac{100}{n} \sum_{n} \left| \frac{NDSI_{observed} - NDSI_{predicted}}{NDSI_{observed}} \right| \quad (3)$$

Predictions were made by considering a rolling series of two consecutive images in a scene. For each pair we extracted the motion vectors and generated the next image as depicted in Sect. III.

A summary of results can be seen in Fig. 11 and Fig. 12.

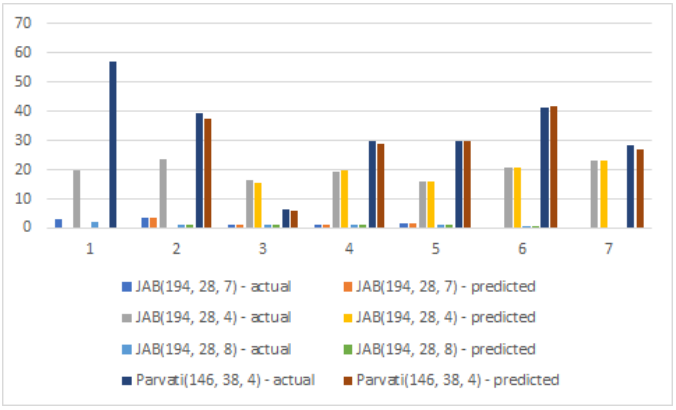

Fig. 11. Comparison between the actual and predicted NDSI values in percentage of covered areas.

C. Jungfrau-Aletsch-Bietschhorn (194, 28, 4)

The motion generated image for this path and row can be seen in Figure 13. This scene was captured in April, when temperatures are still low and snow fall fluctuates a lot resulting in a low signal to noise ratio. This pattern proved to yield inconsistent results for the generated images. We obtained a MAPE of 2.12%.

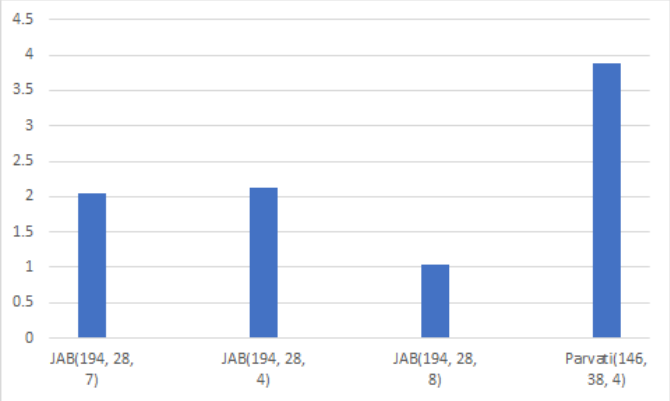

Fig. 12. MAPE comparison between the actual and predicted NDSI values.

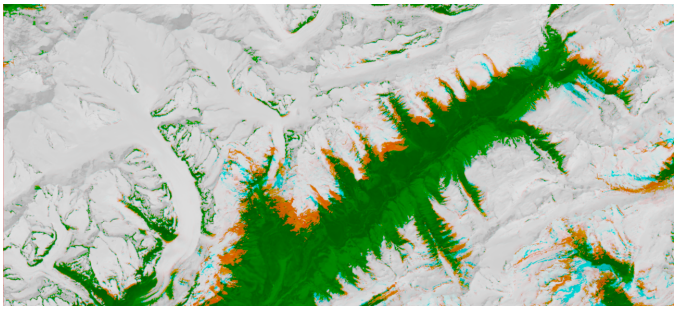

Fig. 13. Overlapped motion generated NDSI for the Jungfrau-Aletsch-Bietschhorn glacier, at path 194, row 28, captured in April.

D. Jungfrau-Aletsch-Bietschhorn (194, 28, 7)

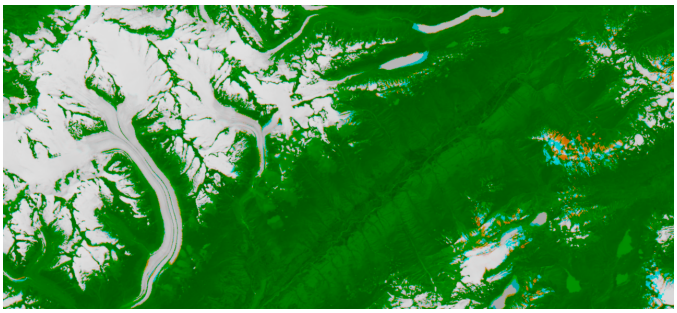

Fig. 14. Overlapped motion generated NDSI for the Jungfrau-Aletsch-Bietschhorn glacier, on path 194, row 28, captured in July.

Figure 14 shows the motion predicted NDSI image afferent to this path, row and month pair. As it can be seen in Figure **??**, the scene was captured in July, hence the more consistent snow fall. These results yield a better estimation of movement than the one obtained in early spring with a MAPE of 2.04%.

E. Jungfrau-Aletsch-Bietschhorn (194, 28, 8)

The result of the experiment is visible in Figure 15. The image was captured during August, which in this case has the least amount of snow fall. The result for this data set was the most reliable, as one can observe since the moraines are clearly exposed, a fact also demonstrated by the MAPE value of only 1.04%.

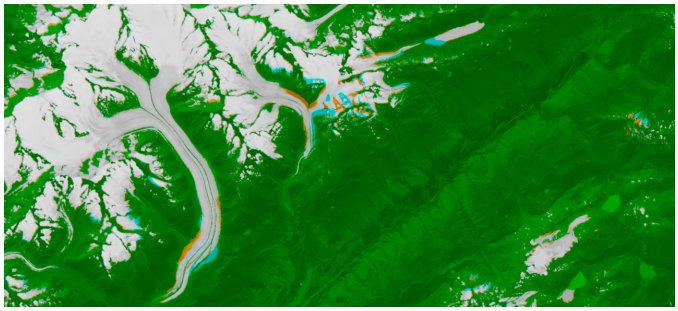

Fig. 15. Overlapped motion generated NDSI for the Jungfrau-Aletsch-Bietschhorn glacier, on path 194, row 28, captured in August.

F. Parvati (146, 38, 4)

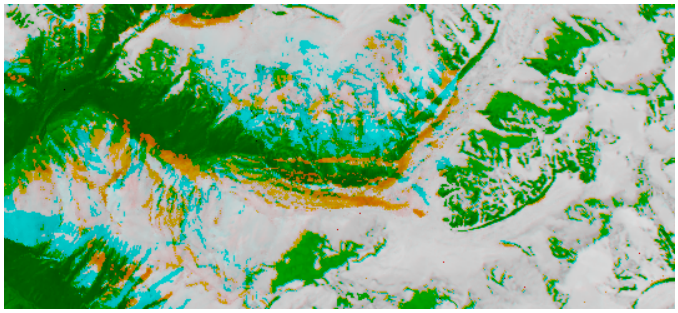

Fig. 16. Overlapped motion generated NDSI for the Parvati glacier, path 146, row 38, captured in April.

Here we can see again that the high snow fluctuation yields in big differences between the predicted image and the actual one, as it can be observed in Figure 16. The MAPE in this case is 3.87% Given that the Parvati region is located at a higher altitude than the Jungfrau-Aletsch-Bietschhorn one, its snowfall is higher. This can also be seen while analysing the snow coverage values for the values from Figure 11, where the maximum snow fall for Parvati is around 57% (2014) and the one for Jungfrau-Aletsch-Bietschhorn is around 23% (2016).

G. Performance

For the performance evaluation we are using a machine which has an Intel i7-7700HQ CPU, with 24 GB of RAM. The images were stored remotely and mounted through NFS.

Downloading the dataset of two glaciers which have been used for the experiment took around two hours on my machine, while their querying took around half an hour. For Parvati there were 185 entries, while for Jungfrau-Aletsch-Bietschhorn there were 109. The processing and caching of all datasets took around 8 hours which sums up to around 1 minute and 30 seconds per scene.

V. Conclusions and Future Work

The best and consistent results were obtained for scenes which were acquired between April and September, due to

lower snow fall and better glacier exposure. We have found that scenes which are taken during the winter months are prone to high errors due to higher snow coverage which creates high fluctuations and outliers. In [4] (Parvati), only September was taken into consideration when conducting the experiments, mainly because it is the month which has the lowest recorded snow coverage of the area. Also, in [5], the experiments were performed on seasonal organized data and the scenes with higher snow coverage proved problematic as well. Another factor which influenced the results is cloud coverage, since even if the shortwave infrared band filters out most of the clouds found in a scene, still there are some cases when the coverage is very dense and the land underneath cannot be analysed. Since the terrain is not visible, the glacier pixels cannot be properly extracted, resulting in erroneous snow coverage and distorted motion generated images at the locations where the clouds have appeared or disappeared.

During the process of implementing and testing the solution multiple problems were met. One of them was that even though the number of available aerial images collected in the Landsat 8 archive are high most of them have very high cloud coverages. Filtering out results which have lower percentages led to datasets with low number of entities. In turn this meant that we could not test our results on longer time series. As clouds interfere both with extracting glacier pixels from a scene and analysing motion we suggest for future work two approaches on:

1) Harvesting images provided by **multiple satellites**, such as Sentinel and older versions of Landsat, as proposed in [6]. Even if these satellites use other types of sensors for Earth observation, by matching the wavelengths of their bands we could organize and pair their datasets, resulting in time series with more entities and more scenes with lower cloud coverage. By only using data provided by the Landsat 8 satellite, even with an allowed cloud coverage of just 20% (as we have used for our experiments) the average length of a $path, row, month$) dataset was around 15 images;

2) Implementing **cloud mapping algorithms** so that clouds could be detected and trimmed out before the calculation of the normalized snow difference index and optical flow extraction. Our results do not consider the difference between snow pixels and cloud ones.

Another possible improvement is to change the way in which the coordinates of the pixels from the motion generated image are calculated. **Time series forecasting models** could be applied on the distance vectors generated by optical flow for each pixel over time in order to generate future entries. This could be done with statistical methods such as the autoregressive integrated moving average (ARIMA). However, doing this forecast over each pixel of a high resolution image would take a lot of processing power. Furthermore, a larger $path, row, month$ dataset would be needed for such an analysis. Preliminary work on the Jungfrau-Aletsch-Bietschhorn glacier's snow coverage by using linear regression and ARIMA has demonstrated that promising results with 95% confidence level for ARIMA, and a good performance by the linear regression for short term forecasts ($R^2 = 0.92$ – coefficient of determination – for 3 year forecasts) can be achieved. As an exercise we predicted (by using a 0.65 NDSI threshold to avoid seasonal snow) that the glacier would completely disappear by 2800 if current trends remain stable.

One of the main slow downs in developing and using the application was the slow processing time due to the large data files. **Migrating** the processing unit to a **cloud** infrastructure and hiring a machine with a CPU which has a high number of cores and better performance could solve this issue. However, in this case we would have to consider the amount of time needed for moving the large scene files to the machine on cloud. A solution would be to move the dataset before starting the processing and make sure that they are on the same machine. However, cloud storage can get very expensive and the trade off between processing time and expenses could prove too unbalanced.

REFERENCES

[1] T. Greene and P. Jacobs. (2021) Nasa report on 2020 average temperatures. [Online]. Available: https://www.nasa.gov/press-release/2020-tied-for-warmest-year-on-record-nasa-analysis-shows

[2] G. Jouvet, M. Huss, M. Funk, and H. Blatter, "Modelling the retreat of grosser aletschgletscher, switzerland, in a changing climate," *Journal of Glaciology*, vol. 57, no. 206, p. 1033–1045, 2011.

[3] S. H. Winsvold, A. Kääb, and C. Nuth, "Regional glacier mapping using optical satellite data time series," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 9, no. 8, pp. 3698–3711, 2016.

[4] S. Tak and A. K. Keshari, "Investigating mass balance of parvati glacier in himalaya using satellite imagery based model," *Scientific Reports*, vol. 10, no. 1, p. 12211, Jul 2020. [Online]. Available: https://doi.org/10.1038/s41598-020-69203-8

[5] A. E. Racoviteanu, K. Rittger, and R. Armstrong, "An automated approach for estimating snowline altitudes in the Karakoram and eastern Himalaya from remote sensing," *Frontiers in Earth Science*, vol. 7, p. 220, Sep. 2019.

[6] S. H. Winsvold, A. Kääb, C. Nuth, L. M. Andreassen, W. J. J. van Pelt, and T. Schellenberger, "Using sar satellite data time series for regional glacier mapping," *The Cryosphere*, vol. 12, no. 3, pp. 867–890, 2018. [Online]. Available: https://tc.copernicus.org/articles/12/867/2018/

[7] H. Wang, R. Yang, X. LI, and S. CAO, "Glacier parameter extraction using landsat 8 images in the eastern karakorum," *IOP Conference Series: Earth and Environmental Science*, vol. 57, p. 012004, feb 2017. [Online]. Available: https://doi.org/10.1088/1755-1315/57/1/012004

[8] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis*, J. Bigun and T. Gustavsson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370.

[9] H. Yan, G. Zhou, and X. Lu, "Comparative analysis of surface soil moisture retrieval using VSWI and TVDI in karst areas," in *International Conference on Intelligent Earth Observing and Applications 2015*, G. Zhou and C. Kang, Eds., vol. 9808, International Society for Optics and Photonics. SPIE, 2015, pp. 37 – 49. [Online]. Available: https://doi.org/10.1117/12.2207397

[10] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, p. 381–395, Jun. 1981. [Online]. Available: https://doi.org/10.1145/358669.358692

[11] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, p. 674–679.

[12] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2