COMPUTER VISION CHALLENGE

SS 2021

Group Member (G06):

Shun Hu

Yuqicheng Zhu

Kaichen Ni

Peng Xie

Wantao Li

CONTENTS

INTRODUCTION

In the last decades, human activities impact the environment in many ways: overpopulation, deforestation, burning fossil etc. These changes have caused climate change which also leads to the shrinking of the glaciers and temperature increase. We as human beings must pay attention to climate change.

But how we can know more about the degree of the changes caused by climate change. One way is to compare places from different periods which can give users a more intuitive feeling. So, we can use satellite imageries to effectively detect and analyze it. It has wide applications in meteorology, oceanography, biodiversity conservation and geology.

Satellite data provide more than half of the authoritative information about 50 key climate change variables. These insights include satellite radar altimeters, which measure the distance between satellites and the earth's surface and provide us with accurate information about sea level. Atmospheric chemistry and greenhouse gases such as methane can also be measured by satellite. At present, about 162 satellites in orbit are used to measure various indicators related to climate change.[1]

The new generation of satellites improves optical and temporal resolution, thereby improving weather forecasting, climate modeling, and access to real-time detailed information. In the next five years, many new satellite missions will be launched, including EUMETSAT second generation[2] polar orbit satellite, third generation meteorological satellite and China satellite.

[1] https://zhuanlan.zhihu.com/p/112486119
[2] https://www.eumetsat.int/metop-sg

The purpose of this project is to visualize the changes from the satellite images, which were taken at different time and from different points of view.

PROCESS

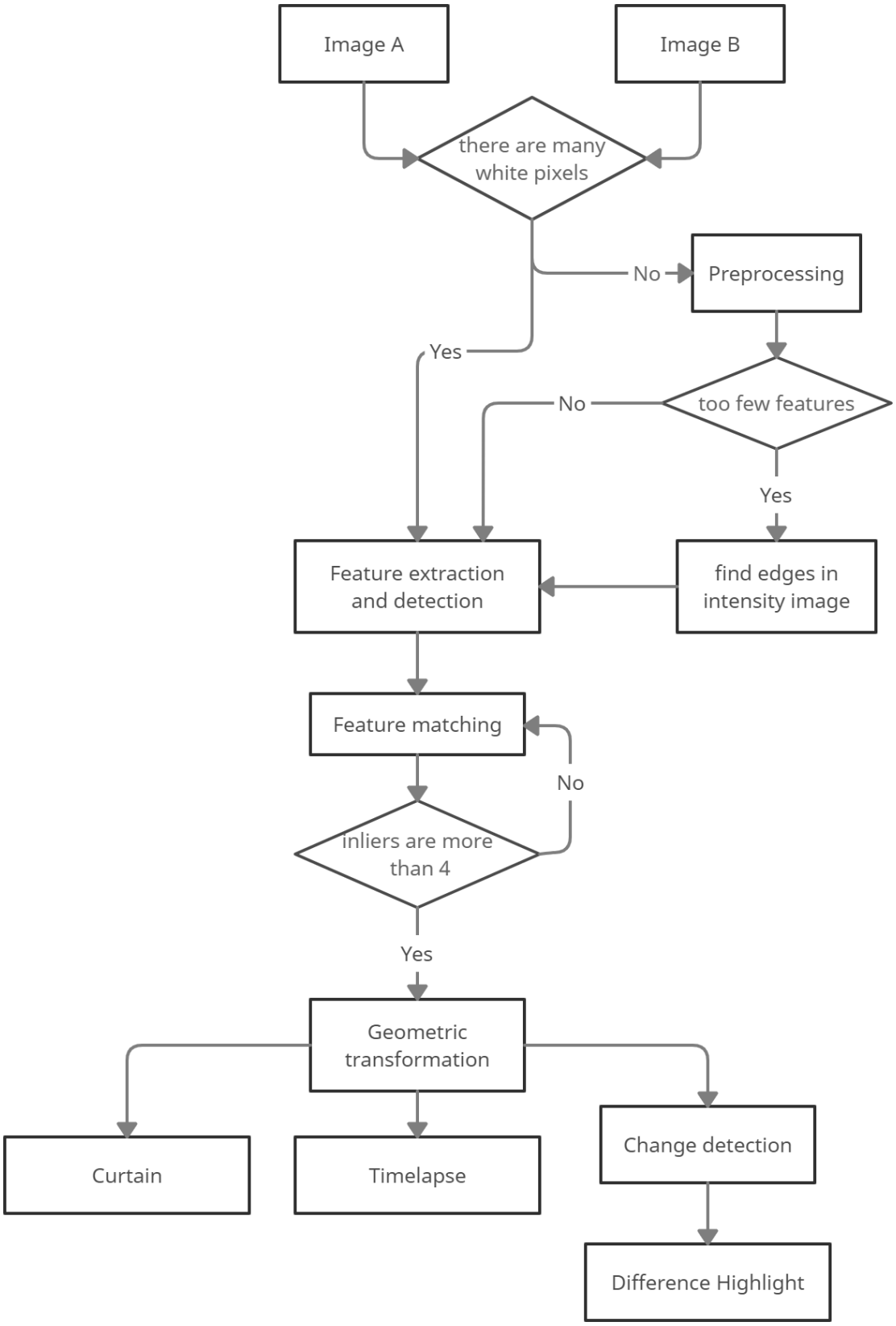


Image A
Image B
there are many white pixels
No
Preprocessing
Yes
too few features
No
Yes
Feature extraction and detection
find edges in intensity image
Feature matching
No
inliers are more than 4
Yes
Geometric transformation
Curtain
Timelapse
Change detection
Difference Highlight

PREPROCESSING

Due to the interference signal in the image preprocessing for the sources turns out to be very necessary. In particular we adjust the intensity of pixels in order to make it more suitable for SURF algorithm, as SURF is sensitive and unstable to illumination.

Image Processing Toolbox[3] helps us to perform the preprocessing of the images such as image enhancement and noise reduction without long and tedious calculations (See Figure 1).

HISTOGRAM EQUALIZATION[4]

Histogram flattening is to transform an image with known gray probability density distribution into a new image with uniform gray probability density distribution. The result is to expand the dynamic range of pixel values, so as to enhance the overall contrast of the image.

FUNCTION DESCRIPTION

$$J = histeq(I)^{5}$$

The input I is the original image. The gray image I is transformed so that the histogram of the output gray image J has 64 bin and is approximately flat. The objective of this method is to adjust the contrast of the image such that the histogram of the image can match the histogram derived from a reference image.[6]

[3] https://de.mathworks.com/help/images/index.html?s_tid=CRUX_lftnav
[4] https://ieeexplore.ieee.org/abstract/document/4266947
[5] https://de.mathworks.com/help/images/ref/histeq.html
[6] https://en.wikipedia.org/wiki/Histogram_equalization

HISTOGRAM MATCH

Histogram matching, also known as histogram specification, is an image enhancement method that transforms the histogram of an image into a histogram of a specified shape. That is to say, the histogram of one image or region is matched to another image. Make the hues of the two images consistent. It can match the histogram of single band image, and it can also match multi band image at the same time. Before comparing two images, the histogram form should be consistent.

FUNCTION DESCRIPTION

$$J = imhistmatch(I, ref)$$[7]

It transforms the 2-D grayscale or true color image I returning output image J whose histogram approximately matches the histogram of the reference image ref. In other words, the aim of this step is to adjust histogram of 2-D image to match histogram of reference image.

PREFILTERING

The most important reason that we filter the images is the noise reduction. In our project, we have used a 40th order FIR low-pass prefiltering to remove noise and sharpen the images.

FUNCTION DESCRIPTION

$$image\_prefiltered = prefilterlowpass2d(double(image), kernel)$$

The inputs are the image in double-precision arrays and the FIR filter kernel. And the output is the image after the prefiltering.

[7] https://de.mathworks.com/help/images/ref/imhistmatch.html

Figure 1: top left: original image, top right: image after histogram equalization, lower left: image after histogram match with the reference image, lower right: image after FIR low-pass filtering

EDGE DETECTION

In the case that the dataset has relatively large changes, especially in the years with a relatively large year span, there are fewer feature points that can be matched between the two pictures. We decide to extract the references such as rivers as our feature edges, so that we can match the feature points on the edges (see Figure 2).

FUNCTION DESCRIPTION

$$BW = edge(I, method, threshold)$$[8]

This function returns all edges BW on the image I that are stronger than the threshold. We use 'Roberts' as our edge detection method, which has almost none approximation to the derivative.

[8] https://de.mathworks.com/help/images/ref/edge.html

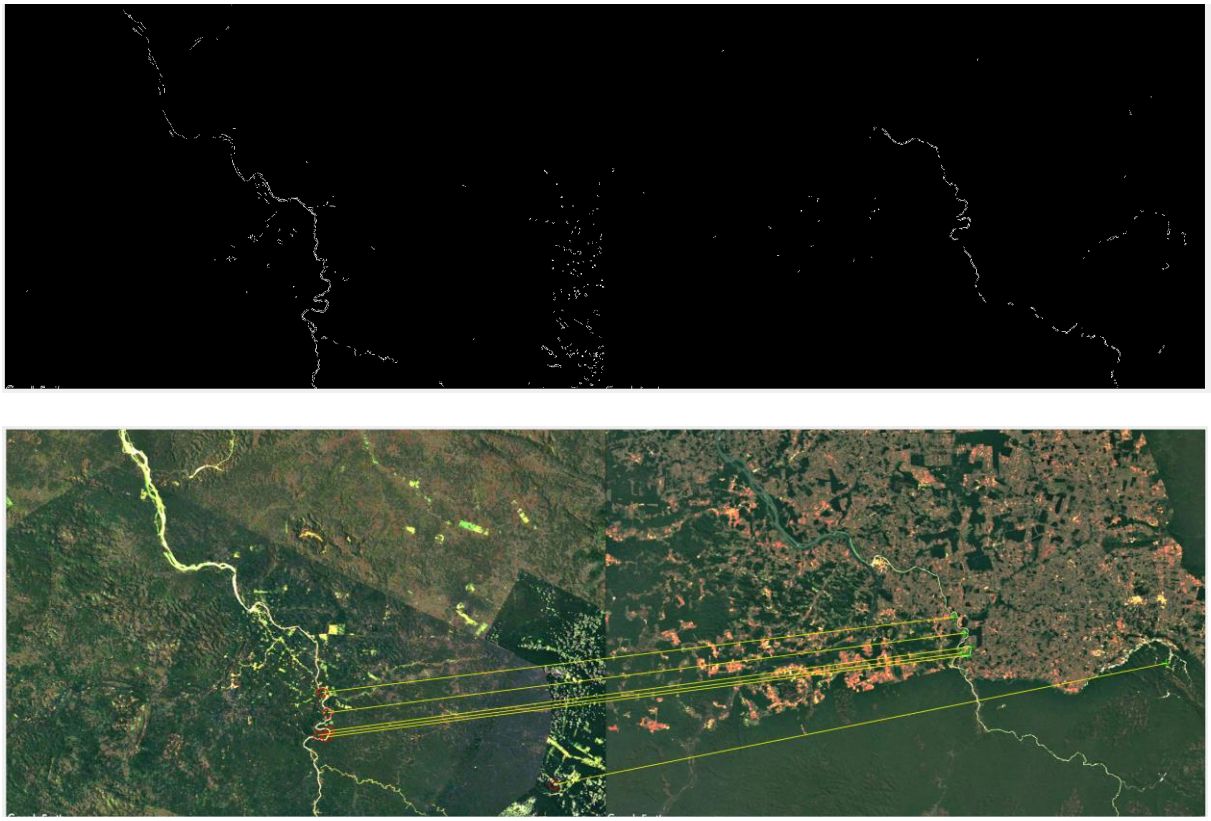


Figure 2: top left: Rainforest 1985_12 after edge detection, top right: Rainforest 2020_12 after edge detection, below: feature matching between Rainforest 1985_12 and Rainforest 2020_12

ALGORITHM

For the Feature Detection and Extraction, we have used the Computer Vision Toolbox[9], which provides algorithms such as SURF, SIFT, which helps us to perform the image registration only with several command.

FEATURE DETECTION

Speeded up robust features, short SURF[10], is the most common method to detect and describe the scale- and rotation-invariant features[11]. It is partially stimulated by the scale-invariant feature transform (SIFT) descriptor. SURF outperforms SIFT in respect of repeatability, peculiarity, and robustness, so far can be computed much faster than SIFT[12].

FUNCTION DESCRIPTION

$$points = detectSURFFeatures(I, Name, Value)^{13}$$

as shown above, the output $points$ is a SURF Points object, which are detected in the 2D grayscale image I. We have set the metric threshold as 500, the number of octaves as 5 and the number of scale levels as 6, which has a best performance in our case.

FEATURE EXTRACTION

In computer vision, scale invariant features are introduced. The main idea is that each feature point is accompanied by the corresponding size factor. When we want to match different images, we often encounter the problem of different image scales. The distance of feature points in different images becomes

[9] https://de.mathworks.com/help/vision/index.html?s_tid=CRUX_lftnav
[10] https://people.ee.ethz.ch/~surf/eccv06.pdf
[11] https://en.wikipedia.org/wiki/Speeded_up_robust_features
[12] https://link.springer.com/content/pdf/10.1007/11744023_32.pdf
[13] https://de.mathworks.com/help/vision/ref/detectsurffeatures.html

different, and the object becomes different size. If we modify the size of feature points, it will cause intensity mismatch. In order to solve this problem, a scale invariant surf feature detection method is proposed, in which the scale factor is added to the feature points. Surf is similar to sift algorithm, SIFT algorithm is more stable, more feature points are detected, but the complexity is higher, while surf algorithm needs simple operation, high efficiency and short operation time. It has following steps:

1. Construction of Hessian matrix

2. Scale space generation

3. Using non maximum suppression to preliminarily determine the feature points and accurately locate the feature points

4. Select the main direction of feature points

FUNCTION DESCRIPTION

$$[features, validPoints] = extractFeatures(I, points)^{14}$$

The output of the function is the extracted feature vectors and their corresponding locations, and the input is the gray-scaled image I.

This function procures the descriptors from pixels surrounding an interest point.

[14] https://de.mathworks.com/help/vision/ref/extractfeatures.html

FEATURE MATCHING

Feature matching is based on the feature descriptor. The feature descriptor mentioned above is usually a vector. The distance between two feature descriptors can reflect the degree of similarity, that is, whether the two feature points are the same. With the method of calculating the similarity of descriptors, how to find the most similar feature points in the set of feature points is the matching of feature points.

In our project, we found that the randomness of matching has a big impact on the robustness. To reduce the randomness, we choose 'Exhaustive' mode instead of 'Approximate' to be our Matching method, which compute the pairwise distance between the feature vectors.

FUNCTION DESCRIPTION

$$indexPairs = matchFeatures(featuresA, featuresB, Name, Value)$$[15]

The inputs $featuresA$ and $featuresB$ are the feature matrix obtained from $extractFeatures$. The output $indexPairs$ is the index of the corresponding features.

We have set the parameter '$Unique$' to true, the function will eventually return the unique matching between $featuresA$ and $featuresB$ (note the order of features). The specific process is a forward backward matching. After matching $featuresA$ and $featuresB$, $featuresB$ and $featuresA$ will be matched in turn, and the best matching will be retained.

The Parameter '$MatchThreshold$' is the Matching threshold, which is a percentage value. To reduce the randomness of the matching we have set the matching threshold to 100.

[15] https://de.mathworks.com/help/vision/ref/matchfeatures.html

TRANSFORMATION MATRIX

Two images with the same content may be out of geometry due to the imaging angle, perspective relationship and even the lens itself.

It presents a different appearance, which brings trouble to the observer or image recognition program. By means of appropriate geometric variation,

we can eliminate the negative effects of these geometric distortions to the greatest extent, which is conducive to our subsequent processing and recognition work.

Focus on the content of the sub image itself, more specifically the object in the image, rather than the angle and position of the object.

Therefore, geometric transformation, as a preprocessing step in other image processing applications, is one of the core tasks of image normalization.

The estimation of the geometric transformation is based on the matching point pairs and there are a significant number of mismatches features among the initial matches. The aim is to find the correct matches, which obey the epipolar geometry.

For this reason, we have used RANSAC (robust estimation by random sampling)[16] for estimating homography in the lecture. The RANSAC has showed successful for robust estimation by reducing the number of correspondences with error below a given threshold in a few steps (see Figure 3):

1. Determine the data model and the number of the necessary parameters k
2. Select k arbitrary data points and calculate the parameters
3. Find the amount of the data points M, which do not exceed the distance d from the line

[16] https://www.moodle.tum.de/pluginfile.php/2827780/mod_resource/content/0/Uebung%20RanSaC.pdf

4. Save the parameters if the neu M is bigger than the old one
5. Repeat the steps above

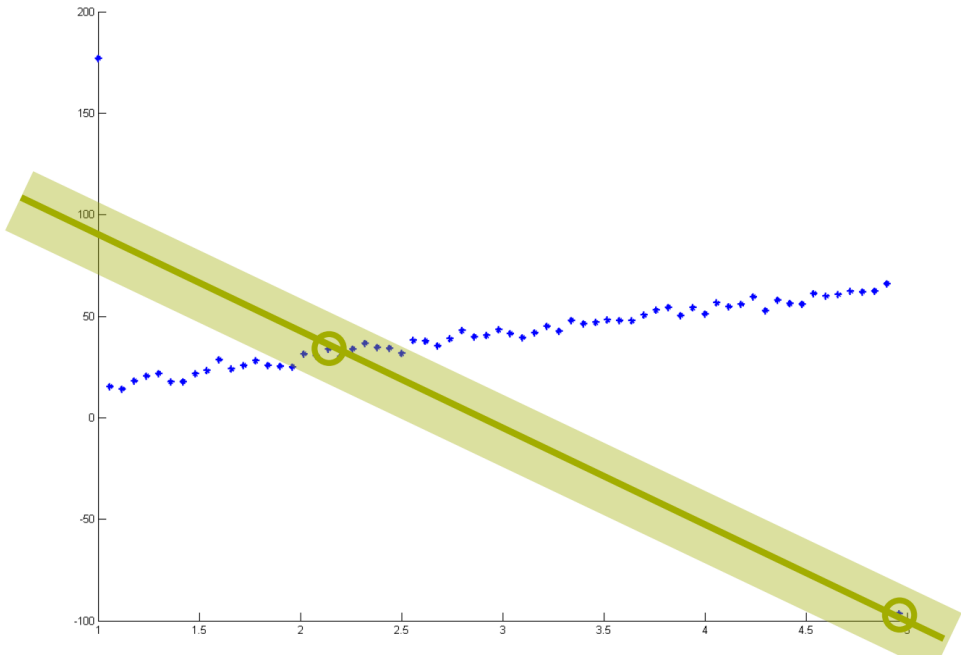


Figure 3 : two arbitrary data points are selected and connected by the green line, the confidence area around the line is defined by the distance d and marked in green

However, there is still some aspects that RANSAC can be improved on. One problem is that RANSAC does not work sufficient robust if the threshold T for considering inliers is set too high.

RANSAC tries to find the minimum of a cost function C to some extent,

$$C = \sum_i \rho(e_i{}^2),$$

where the robust error term ρ is defined as

$$\rho(e^2) = \begin{cases} 0, & e^2 < T^2 \\ constant, & e^2 \geq T^2 \end{cases}$$

e is the MLE error for the point data and e^2 is the appropriate error function for the point data, the ρ is so called the score for each point data.

In other words, if the T is larger than all solutions, the cost would be 0 and all solutions are inliers.

Instead of giving a fixed penalty for the error, MSAC (m-estimator sample consensus)[17] adjusts the error term for all robust estimations as follows:

$$\rho(e^2) = \begin{cases} e^2, & e^2 < T^2 \\ T^2, & e^2 \geq T^2 \end{cases}$$

Thus, the advantages of MSAC are:

1. better cost function
2. more robust because the cost function is determined in terms of the likelihood of inliers and outliers.

FUNCTION DESCRIPTION

$$[tform, inlierIdx] = estimateGeometricTransform2D(matchA, matchB)$$[18]

This function estimates 2-D geometric transformation from matching point pairs. The output $tform$ is the transform matrix and $inlierIdx$ is the index of inliers in vector form.

[17] https://www.sciencedirect.com/science/article/abs/pii/S1077314299908329
[18] https://de.mathworks.com/help/vision/ref/estimategeometrictransform2d.html

GEOMETRIC TRANSFORMATION

After the computation of the transform matrix, we can now apply the geometric transformation for the image registration.

FUNCTION DESCRIPTION

$$outputView = imref2d(size(im\_ref))$$[19]

This function creates the output view of the image, which contains the information about the size and the location of the output image in the global coordinate system.

In this case, we set the output view size equal to the size of the reference image.

FUNCTION DESCRIPTION

$$imgAafter = imwarp(imgA, tform, 'OutputView', outputView)$$[20]

The image $imgA$ is transformed according to the geometric transformation form $tform$. This function returns the transformed image in $imgAafter$ with $outputView$ as the global coordinate system (see Figure 4).

left: image that we want to transform, right: reference image

[19] https://de.mathworks.com/help/images/ref/imref2d.html
[20] https://de.mathworks.com/help/images/ref/imwarp.html

Figure 4: image after transformation

GUI & FUNCTIONALITY

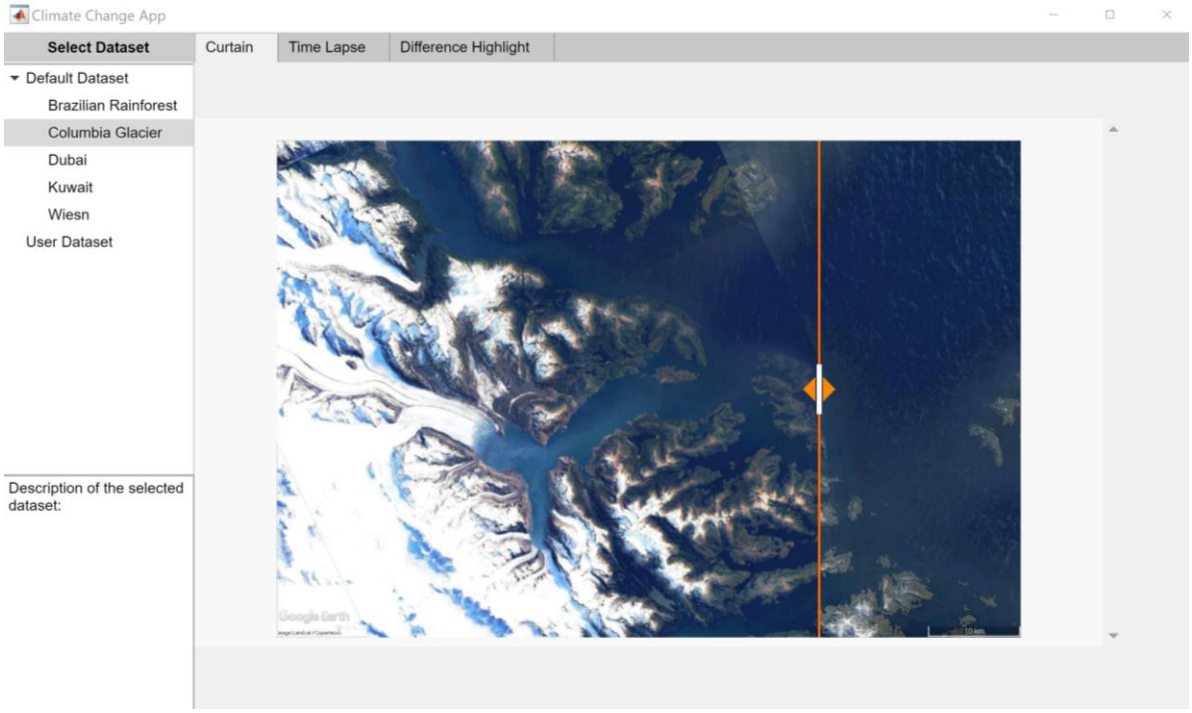
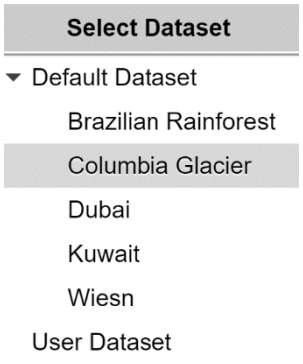


Figure 5: our GUI interface

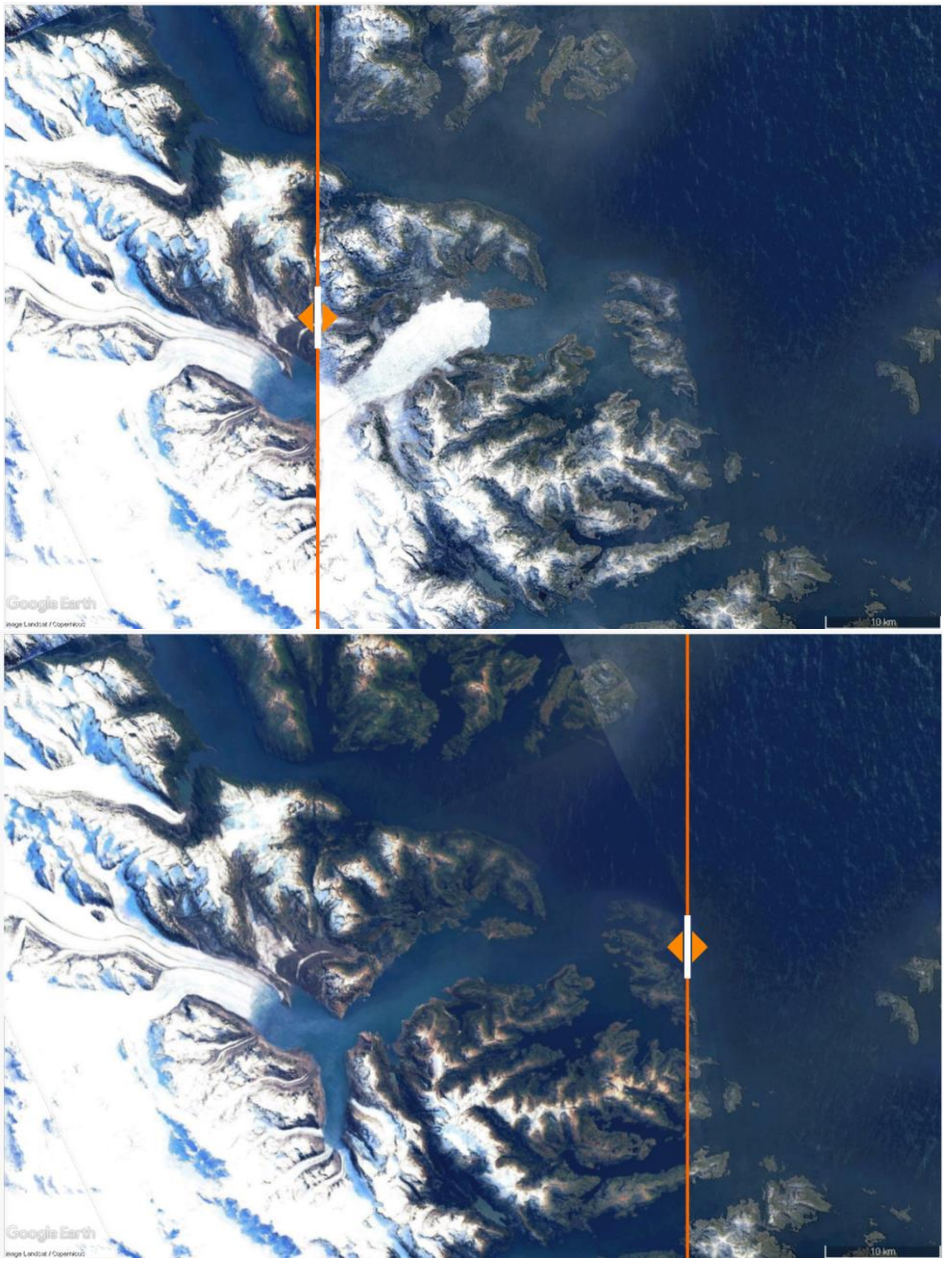
DATASET SELECTION



For dataset selection, we use a standard tree component to display hierarchical list of the datasets. User can select the dataset from the list of available datasets.

DIFFERENCE-CURTAIN

We want to show two different pictures on one picture to give users a more intuitive feeling. Our app has a swiping curtain. In the left and right sides there are two different images so we can easily see the comparison. And through swiping the curtain a better UI experience could be achieved.



TIMELAPSE

we design a timelapse which shows the whole set of pictures. User could select which image they want to review by simply clicking on the button. Also, we set a function which enables automatically playing the whole set of results.

Figure 6: timelapse interface

DIFFERENCE-HIGHLIGHTS

To detect land changes based on intensity, we need to do a median filtering at first, which reduces the band noises and improves the quality of the images significantly.

After the prefiltering, we can detect the changes easily by subtraction of the two images. We ignore the small changes and remove the small objects from the binary image.

FUNCTION DESCRIPTION

$$image\_areaopening = bwareaopen(image, P)$$[21]

This function is from the Image Processing Toolbox. It removes small objects which have fewer connected components than P from binary image.

$$image\_highlighted = showdiff(imgA, imgB)$$

This function shows the difference both demolished and newly created on the images. The inputs are the images that we want to compare and the output is the image with highlights of the differences (see Figure 6).

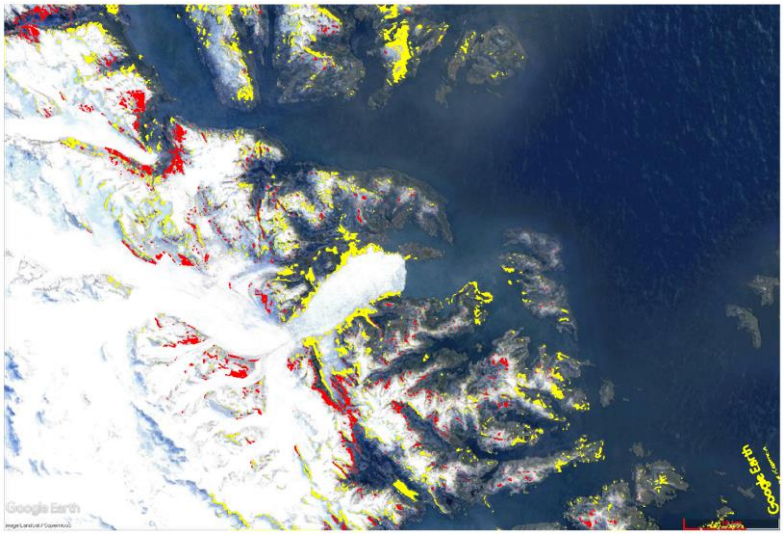
[21] https://de.mathworks.com/help/images/ref/bwareaopen.html

Figure 7: difference highlight interface (Columbia Glacier)

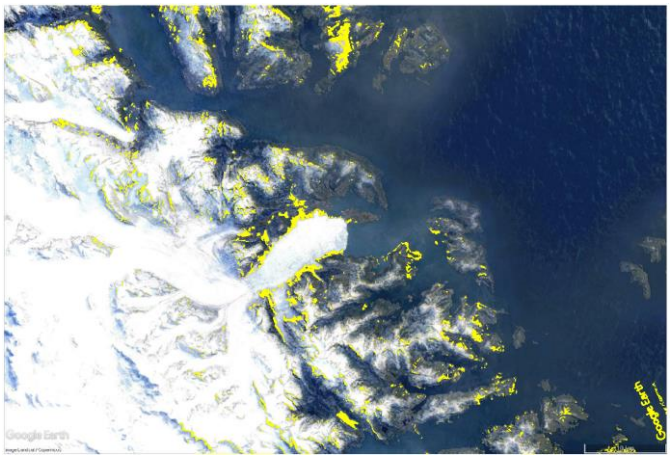
In this interface, user can select two images of different years in one dataset and compare the difference. Furthermore, users can rotate threshold knob to choose the density of the changes.

There are 4 modes for visualization of the difference:

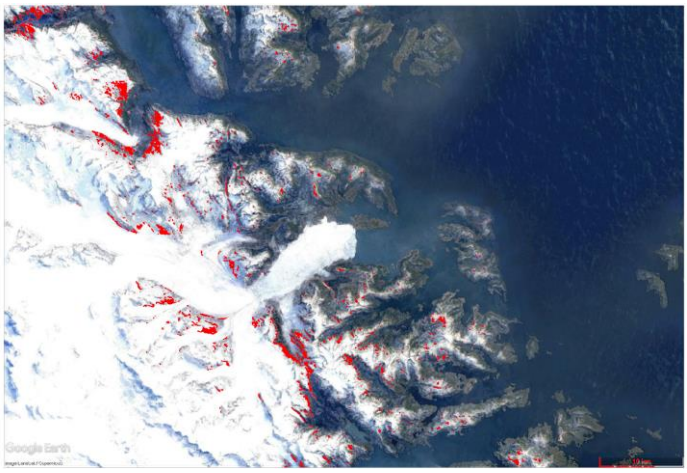
1. In the default mode, it shows both increased (yellow) and decreased (red) changes in the image



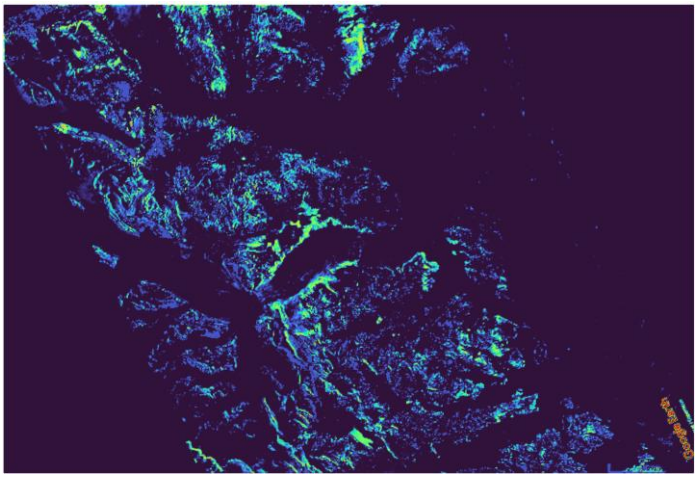
2. Switch the ‘partial’ into ‘on’, the image shows only increased changes in yellow



3. Switch the ‘mode’ into ‘decrease’, then the image shows only decreased changes in red



4. Switch on the ‘heatmap mode’, the image shows different degrees of changes



LIMITATION

Because the “Brazilian Rainforest” has relatively large changes, especially in the years with a relatively large year span, there are fewer feature points that can be matched between the two pictures. And by observing the histogram of the picture, we found that the rainforest pictures have the following problems. Because the picture has most parts with similar colors, it is easy to match the wrong feature points. Secondly, the pixels of the picture are relatively concentrated, and it is difficult to find representative feature points in a variety of concentrated pixels. Therefore, it is inefficient to accurately find the transform matric of the two sets of pictures. In our algorithm, for the first three groups of pictures, the algorithm can extract the rivers in the pictures for matching. The last four sets of pictures are based on naked places. Therefore, for those groups the matching effect is perfect. Unfortunately, this algorithm is not generally versatile. So far, it cannot perform well on the whole dataset. For example, the fourth image, chronologically the image “2000_12”, cannot always be matched with other images. We suppose that the river shape in the fourth image is too subtle to be detected and enhanced by our algorithm, which leads to the match failure.

In a group of Wiesn images, because the city has a lot of shadows, it causes great interference to our feature point matching. In order to find the feature points that can calculate the transform matric. We add loops to the algorithm, and constantly judge whether the found feature points are correct. The final function is that we can accurately match any two sets of pictures. But at the same time, because of the number of cycles, the algorithm requires more time to run, and the running speed is slower. This is the biggest shortcoming of calculating the

transform matric of this group of pictures, and it is also one of the parts that can be optimized in the future.

FUTURE WORK

Considering the runtime of our program and the size of the datasets, we abandoned some processes from our project, which may have better performance and are more precise than our current version but need a Long - Running Process.

In the dataset „Wiesn “, we found that the shadows of the buildings have a big impact on the change detection. The shadows are always detected as changes in multi-temporal images because of the dark color.

A future study can be done to establish an automated and fast method of removing the shadows.

Another important aspect is the method of the change detection. The method we used is subtraction, that shows every single pixel, which has change in intensity. To find the change area that we are interested and not every single intensity change, we can use the principal component analysis (PCA) and k-means clustering[22]. However, this method needs a long runtime and that reduce the quality of our program.

A future study of the change detection using PCA and k-means should be the improvement of the runtime and the reduction of the computation complexity.

[22] https://ieeexplore.ieee.org/abstract/document/5196726