God’s Eye: Landscape Change Detection System

G44: Andrew Madden, Hazem Amami, Sherif Shousha, Essam Hassan, Moncef Ben Hadj Salem

Introduction:

The motivations of using technology as a means to temper negative repercussions of environmental change have become a topic of interest in recent years. In this work, and as part of the Computer Vision Challenge, we provide an application that attempts to focus on that topic. By applying Computer Vision techniques on satellite imagery, the app tries to detect and document changes in the environment over a period of time. Furthermore, by providing a variety of features at the user's disposal, this tool aims to provide a useful analysis given the specific type of the change that is trying to be identified. The following sections lay out in more detail how the individual components of our application work, in the order that they are performed.

Image Preprocessing:

Image enhancement is the process of adjusting images so that the results are more suitable for display or further image analysis. For example, you can remove noise, sharpen, or adjust the contrast of an image, making it easier to identify key features.

In this section the preprocessor function takes input images and performs a set of processes on it to enhance the feature detection and segmentation. These processes are gamma correction, conversion to grayscale, contrast enhancement, and filtering the image to discard the noise.

gamma correction: for this process we use the **lin2rgb()** function to apply gamma correction to linear RGB values.

Input image:

After Gamma correction:

Grayscale conversion: for this process we use the **rgb2gray()** function to convert the rgb image into grayscale image.

Grayscale image:

Contrast enhancement: for this process we use the **adapthisteq()** function to perform contrast-limited adaptive histogram equalization. We use this function to avoid amplifying the noise which might be present in the image.

Image after contrast enhancement:

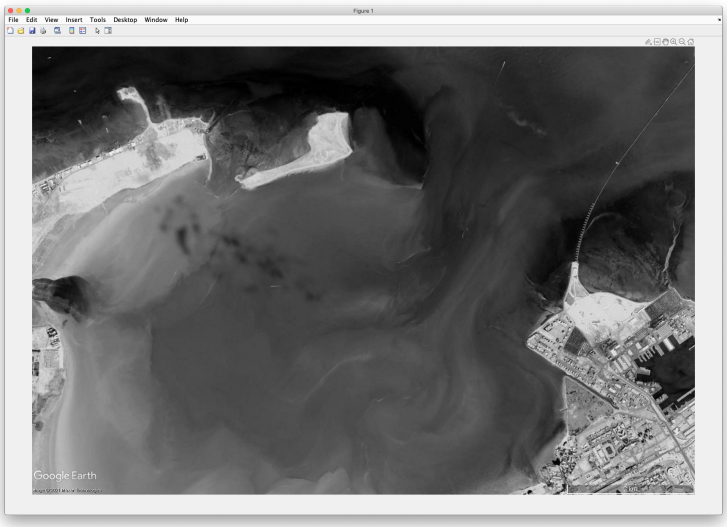
Image filtering: for this process we use the **imflatfield()** function to perform flat-field correction to discard the noise and unwanted shadows.

Final Image after filter:

Registration:

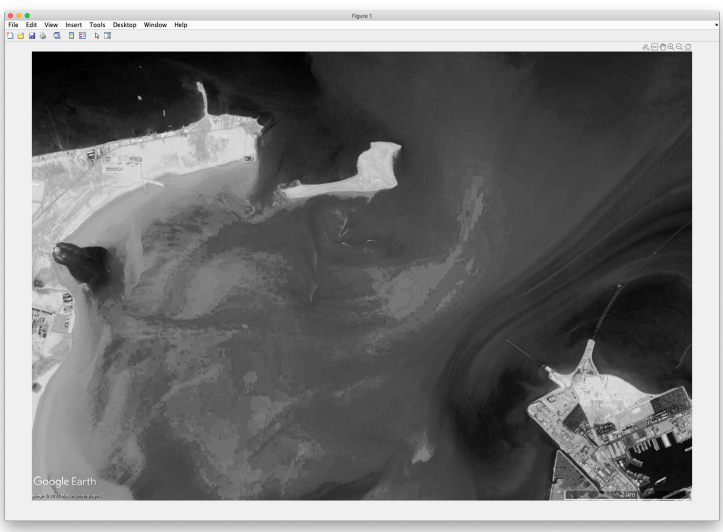
Once the Load and Prepare Dataset is pressed the images are loaded via imread() into matlab. The associated metadata is collected as well. Then, the first image, which is always used as the reference image is directly copied into the output cell array, since no transformation is necessary. To use it as reference for the remaining images, it is then preprocessed depending on the “Image Quality” setting (either just a histeq() lighting adjustment to make it more uniform or the preprocessing step mentioned above). The reference image is then converted to grayscale. This concludes the preparation step of the function.

Reference Image:



Next, in a loop from the 2nd image to the last image the current image is read and preprocessed. This involves either adjusting lighting via imhistmatch() or more extensive preprocessing as mentioned above. Then it is also converted to grayscale.

Imhistmatched and grayscaled input image:

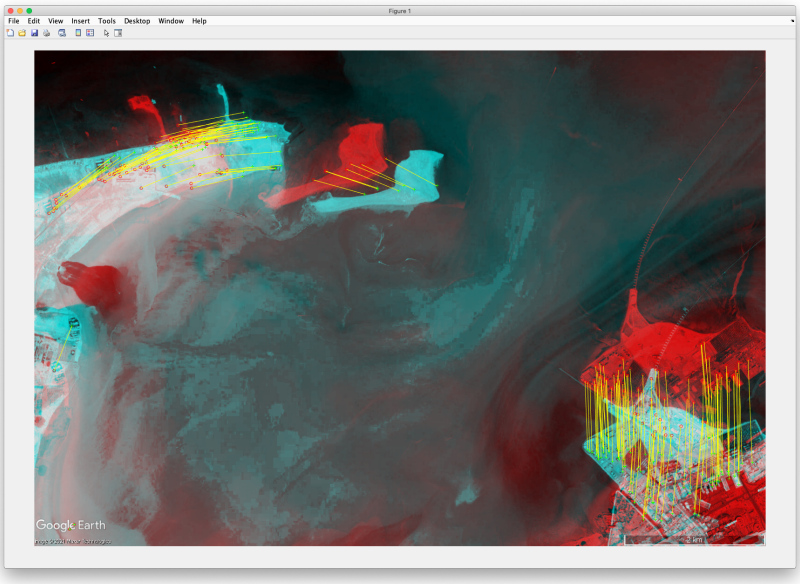


Then the registration function is called. This function takes two grayscale images and attempts to align them. If “Image Quality” was set to the faster option, features are detected via KAZE if robust was selected, SURF is used, which according to this paper and this one is quite robust

but faster than SURF. If the robust option is selected, SURF is used at the expense of slightly longer computation time.

The feature fingerprints are then extracted via BRISK. If a lot of points are extracted, nearest neighbor approximation is used to prevent excessive memory use, otherwise it performs an exhaustive search. The Match Threshold and Max Ratio are slightly relaxed to create more matches for the next step and since RANSAC is applied to reject outliers from the data used to fit the transform.

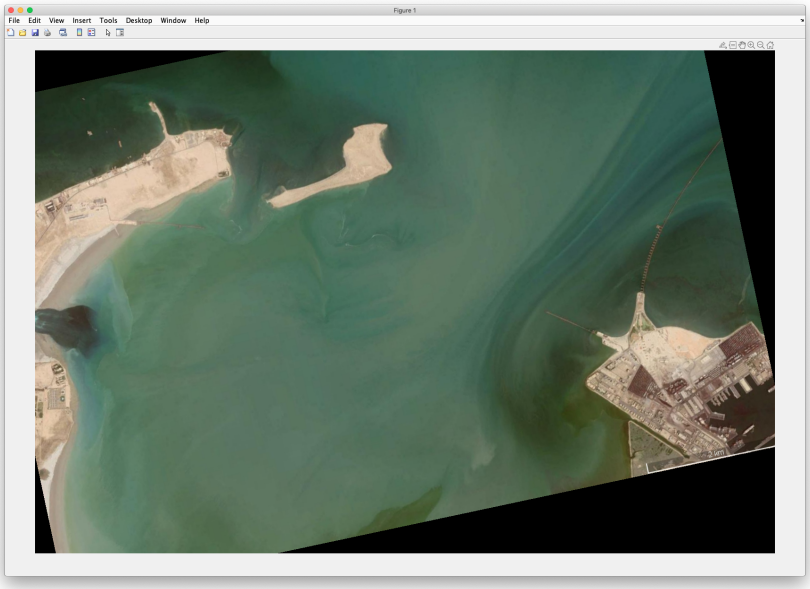
Matched features between reference and matched_image:



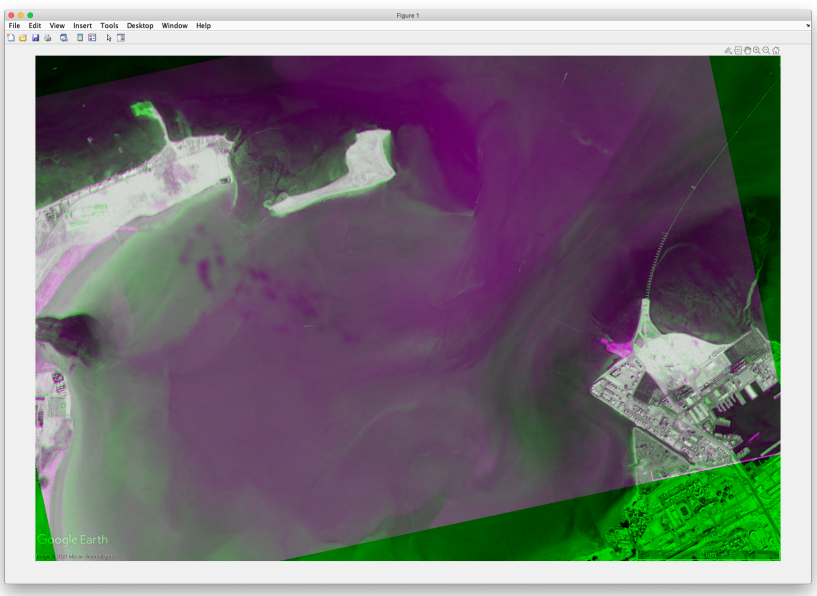
The matched features are then used to try to fit a nonreflective similarity to the matched points. This however requires at least 2 pairs of matched points. If the images are so poorly aligned or no matched points can be found for any other reason (skew, perspective shift etc), an error is thrown and the image is discarded.

The function returns the fit transform to the load data set, which is then applied to the original image. This transformed, color image is then saved in a cell array for later analysis.

Transformed final image output:



Overlay comparison between transformed image and reference image:

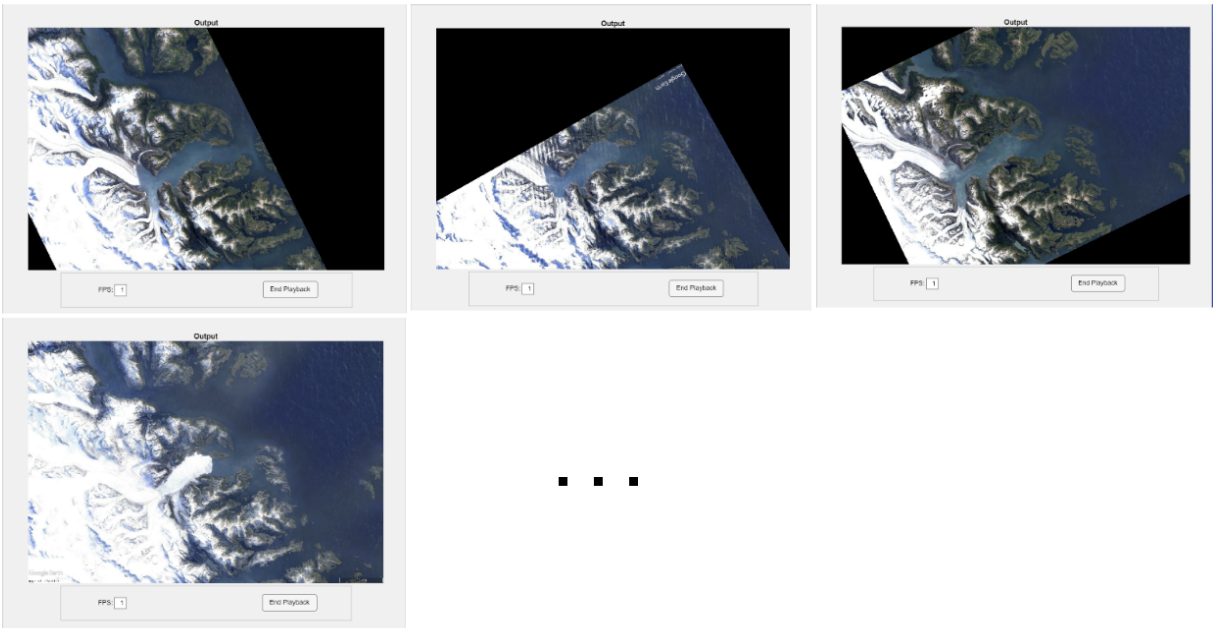


Visualization Types:

After the images have been analyzed, you can choose various ways of evaluating the dataset. Using the list box (8), the user will be able to choose one of three visualization modes and the time scale that is to be examined. If short term is selected Then, once will be able to pick any of the dataset images and compare it with the image directly after it. If long term is selected, the

first and last image in the dataset will automatically be selected to compare.The application will update accordingly:

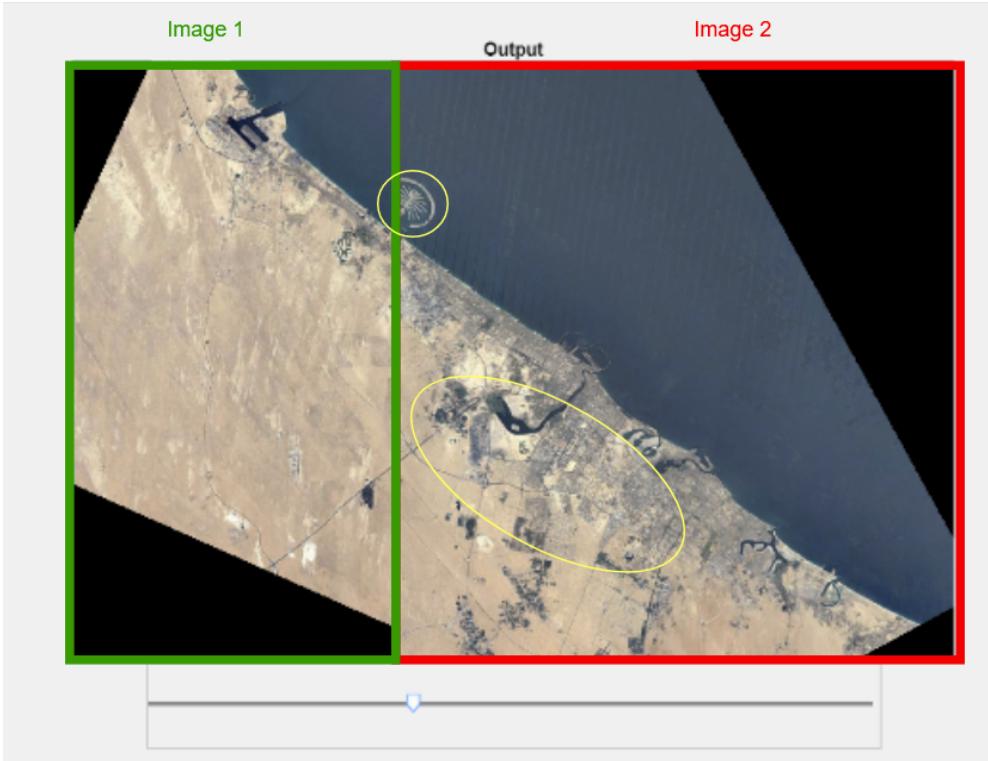
1. [Timelapse:](#) This is the default option. The idea behind this visualization mode is to parse the images of the selected location and perceive the slight change that keeps happening over a period of time. When this mode is selected, a parameters box under the output area will appear and it will be possible to end the time-lapse via “End Playback” button and to change the “FPS” (i.e Frame Per Second) of the output. Per default, the value of FPS is set to 1 frame per second.



. . .

Aligned images from the chosen dataset will appear on the screen with a frequency that depends on the value of “FPS”

2. [Slider:](#) The slider option allows the user to easily navigate between two images of the same location but from different years, this option was implemented in order to make it easier for the user to perceive the shifts in landscape that may occur. When clicking on the slider option, a slider will appear under the image. Moving or dragging the slider will display parts of the images side by side.



As seen in the figure above the slider function allows the user to compare two images from different years. It is clear to see in the example above that the artificially built palm was not present in Image 1 and that the city density clearly increased in Image 2.

To achieve this feature, two images are overlaid. Then, a slider function was implemented, that applies alpha masking to one of the images and depending on the position of the slider part of the image are either rendered fully opaque (100% alpha) or fully transparent (0% alpha).

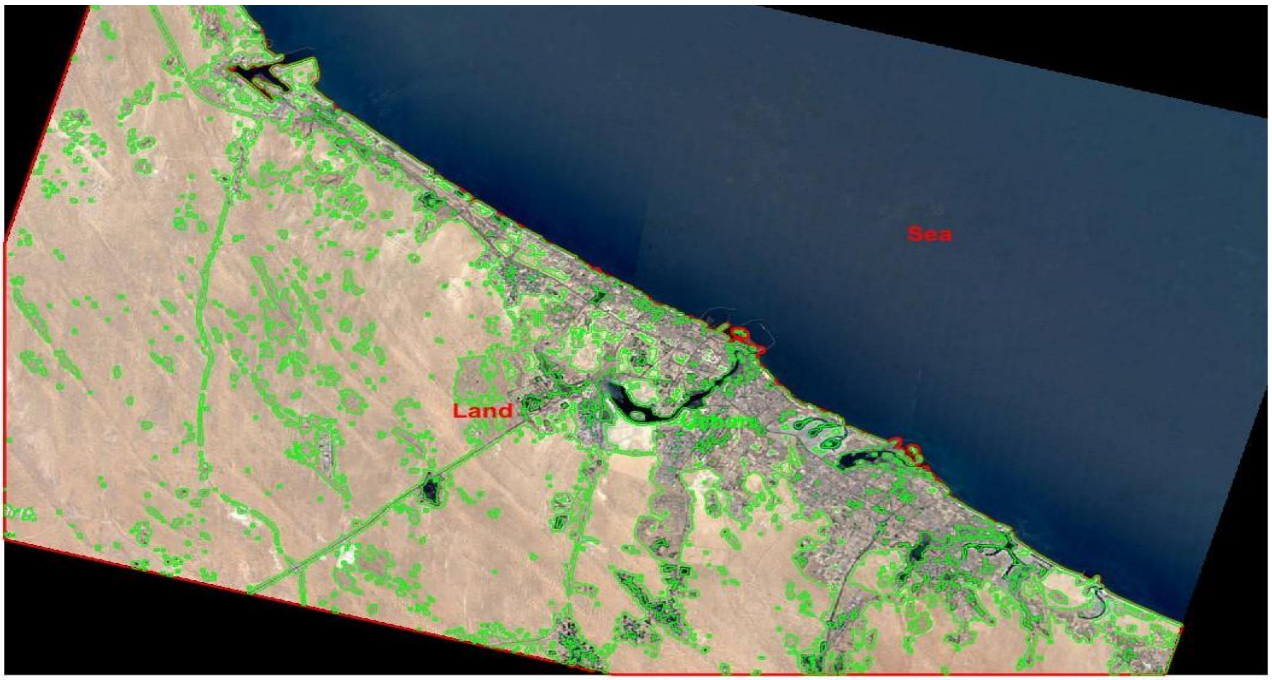
A small drawback of using this method to display images side by side is that moving the slider from i to $i + 1$results in a jump of $x > 1$pixels which results in a small shift between the position of the slider and the border between the two images.

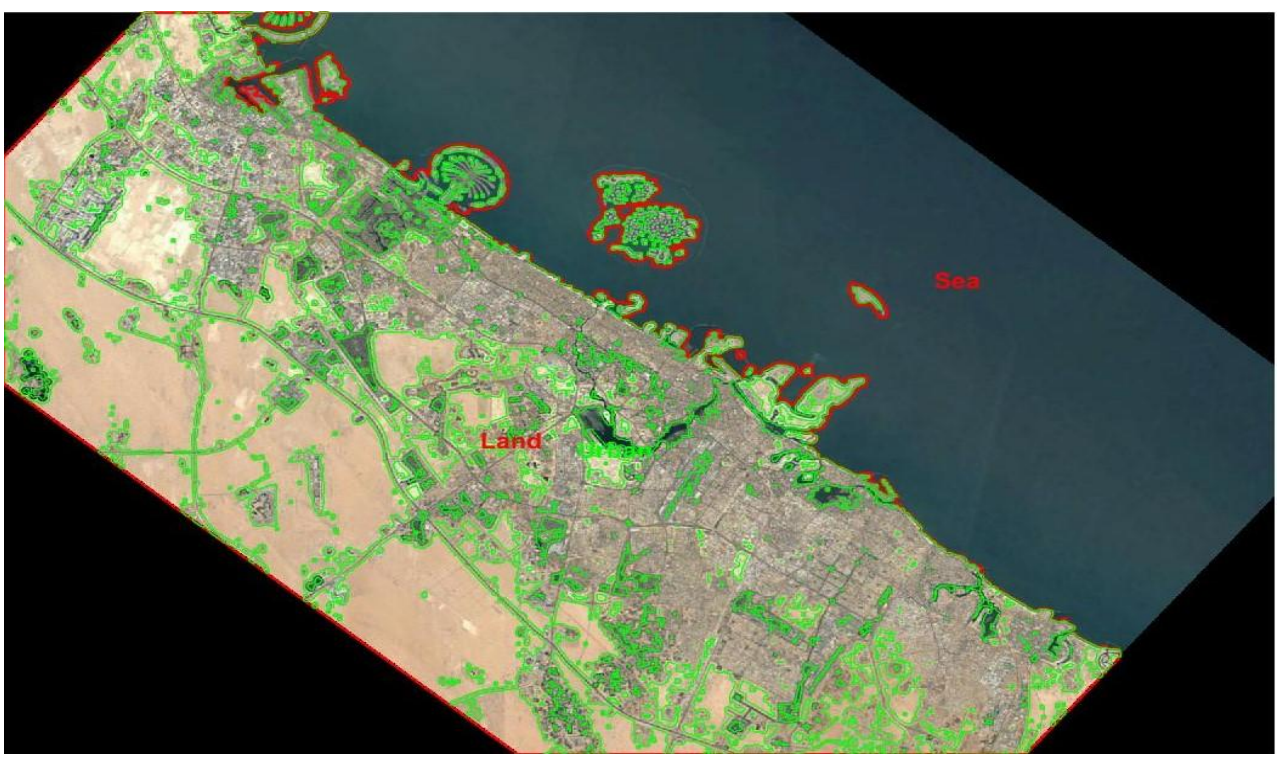
3. [Difference Image:](#) This option will display the difference between the first and last image in the sequence to highlight the difference that occurred in the whole time period. Dark areas signify areas with high amounts of change between the images, while lighter areas signify less change. Furthermore, if the color has shifted over time, the shade of color change will also be displayed.

The difference image is achieved by calculating the SSIM (“Structural Similarity Index Measure”) between two images using the following formula :

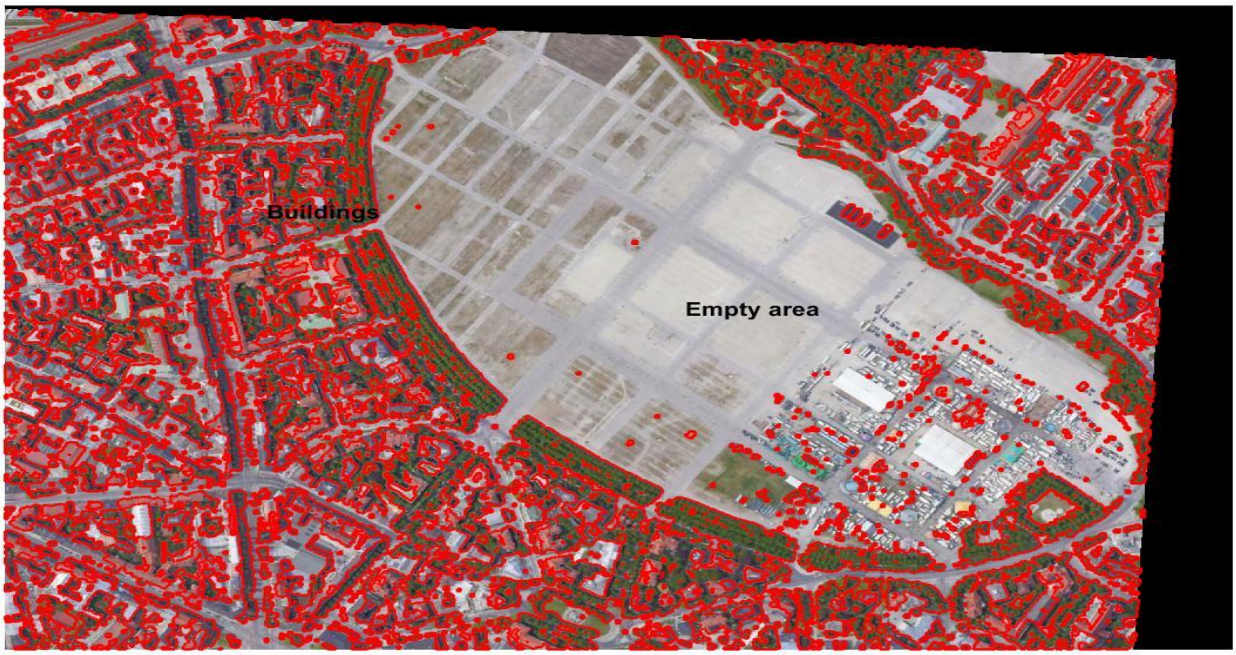
$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

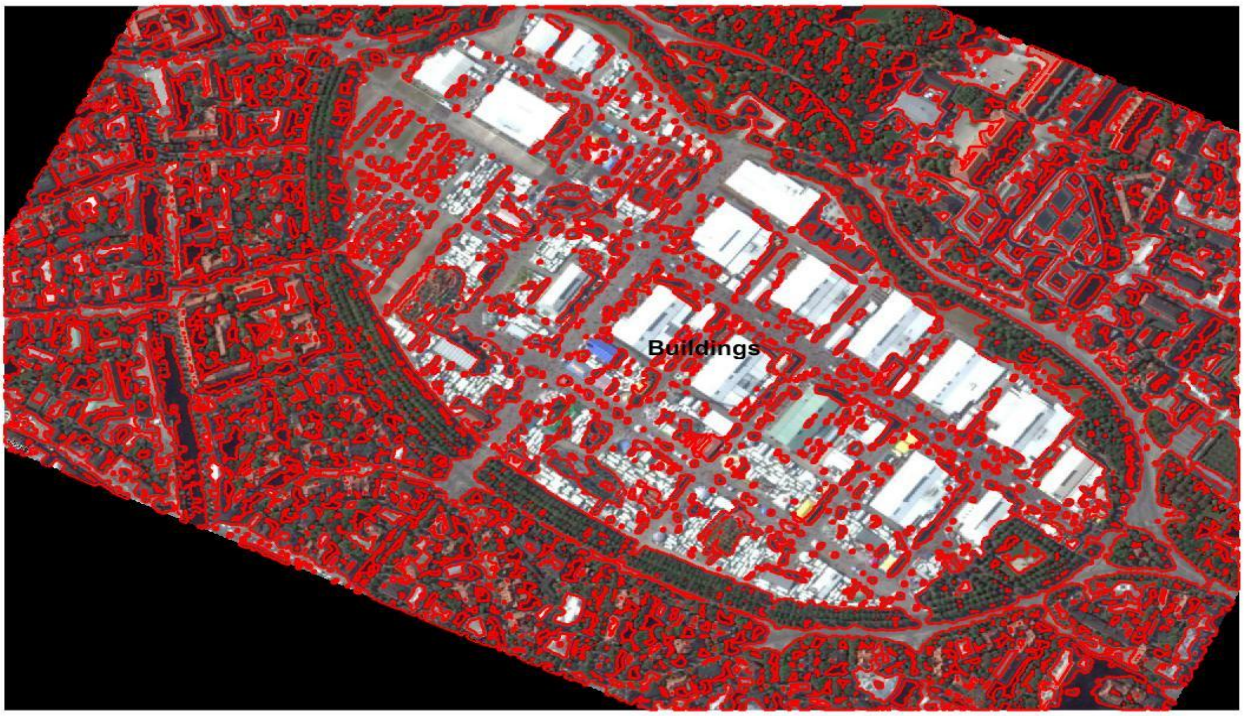
4. Segmentation: The segmentation algorithm detects the different elements in the images and allows the user to recognize this element in the input image. To use this algorithm, the user needs to choose which kind of images he wants to recognize. There are two types, the landscape type, which recognizes the sea and land parts and detects if there is an urban area in the land part or not, as shown in the figure.

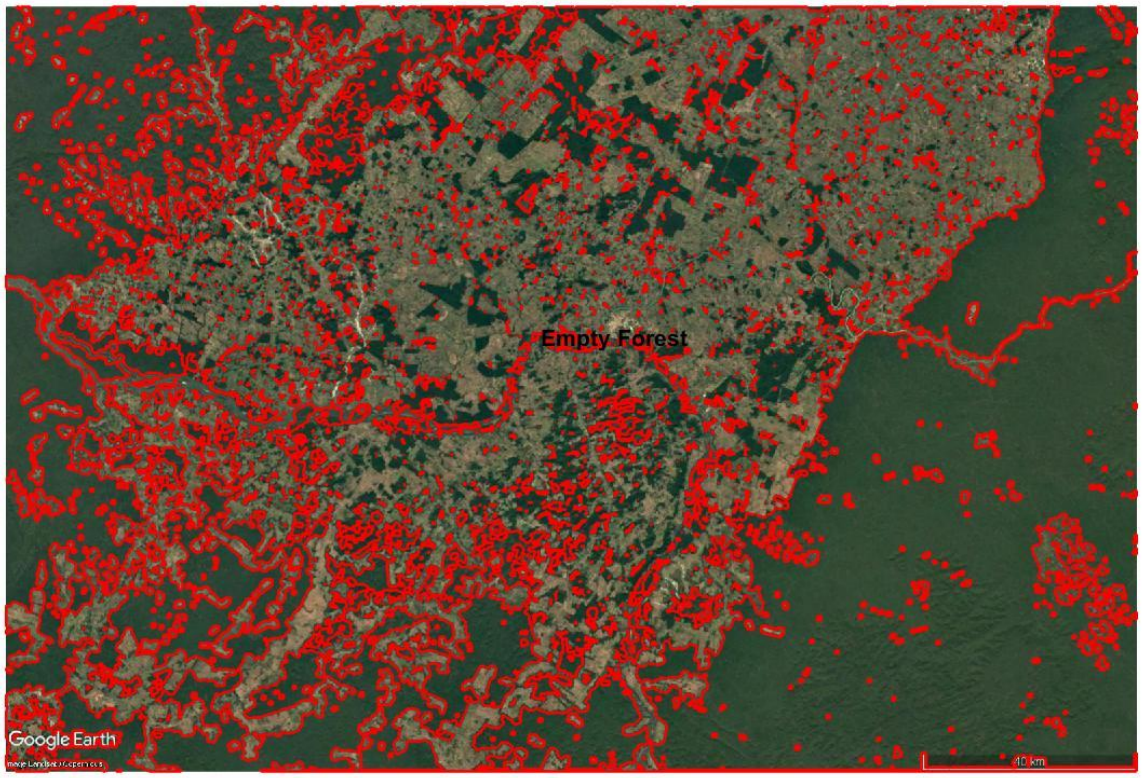




The Urban type recognizes the building and empty areas in the input images. It also detects the green space, and identifies it as a forest area. The algorithm detects the forest image, and the empty regions of the forest are calculated and shown in the output image as shown in the figure.







The algorithm is based on the threshold of different colors, which is used to create masks. In the landscape case, the gray threshold is used to detect the land and sea areas. After that, the buildings' mask calculates the different blobs in the image. Then the algorithm determines if the landscape includes urban and building areas by calculating the area of the maximum blob. Finally, the algorithm creates the appropriate binary mask. In the urban case, the algorithm creates several color masks for the possible colors, such as dark buildings, light buildings, and green areas. The algorithm begins by calculating the maximum blob in the green mask. If the area of the blob is greater than the exact area, the algorithm decides that the current image includes a forest area and creates a mask to detect the difference between the values of the green color. If there is no forest area, the algorithm creates another mask by adding the different color masks and uses the new mask to create a binary mask.

After creating the binary mask in both cases based on the user's purpose, the binary mask is filtered by applying the Gaussian smoothing filter and extracting the mask's noises. After that, the blurred mask is binarized to identify individual blobs by seeing which pixels are connected to each other. Then the algorithm calculates the different boundaries from the binarized image. Finally, the blobs and boundaries are used to identify every part of the image.

Conclusion:

In this report, a detailed explanation of the design of the application has been presented. In addition, despite the complexities that accompany the datasets, the app ensures a variety of features that are useful for the identification as well as the analysis of the change in the environment. This is mainly done by relying on a combination of well-constructed Computer Vision techniques. Nevertheless, it is of importance to mention that there is room for improvement and several challenges have to be tackled with the aim of further enhancing the performance of the tool.