Summary of our practical work over the Hackathon weekend

EscapeWar.net
Jakub Morawski, Adrian Walczak, Victor Lopez, Mariia Utkina, Sarah Cader

5th of October 2023

1 Introduction

In this document we summarize our practical work on designing a prototype of our envisioned **EscapeWar.net** service. As a test case, we analyse satellite images of the Gaza strip from October 2023, where multiple bombings occurred during that month. In 2, we describe how he used *QGIS* to visualize satellite data and figure out how to detect bombing sites. In 3, we describe our Python code which is customized to access data from the Copernicus Hub using the Catalog and Process APIs. Based on this data, we generate danger heat maps, which is described in 4. In 5 we describe our algorithm to find escape routes from danger zones, based on the A^* algorithm. We also dive into other possible data sources for danger assesment in 6. In 7 and 8 we introduce our UI element prototypes - a website and mobile application, respectively.

2 First glimpse into the Sentinel-2 data

As a pilot test to figure out what can be done with the the sentinel-2 imagery, sample images from the last month were extracted using the Copernicus Data Space Ecosystem to define the combination of bands that better suits the need to locate war zone indicators such as bombing sites and affected urban vegetation. Especially for the bombing location finding we found useful the false colour combination of bands, which is commonly used for urban areas prospection. Bombing sites are easily identified in the B12 band and can be tell apart from clouds using the B02 band. The multi-spectral manipulation was carried out with QGIS (Figure 1).

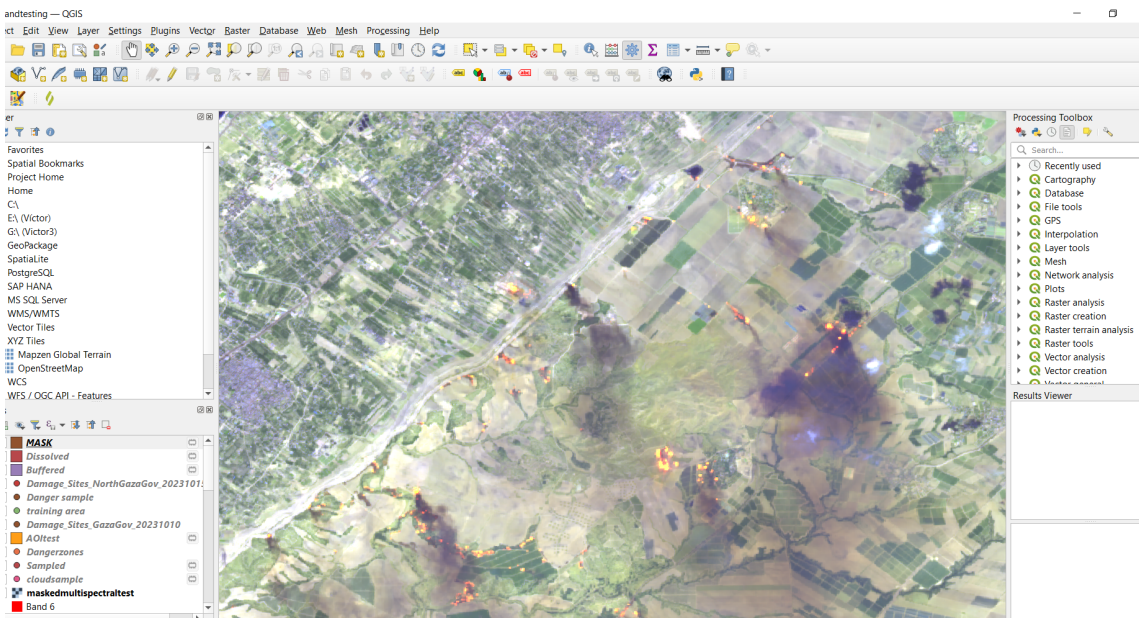


Figure 1: Bombing sites identification using the false colour band combination in QGIS.

3 API approach to data download

Having learned in 2 about how Sentinel-2 data can be useful for us, we proceeded to write a code that could easily access it for any area at any time. The idea was to create a class ***war_zone***, which has functions to check if data is available for a date and download it. This object will store the data as matrices of pixel values. It will also store information about the coordinates of a bounding box.

A challenge with downloading Sentinel data through the Process API is a 2500×2500 pixel constraint, which prevents us from downloading it for an arbitrary big area. To overcome this, we generate a grid of maplets, that fit within the quota, download separately for each and merge this data by concatenating matrices. Figure 2 shows how such a grid of maplets would look like for our area of interest ($31.240207° - 31.570155°N$, $33.955307° - 34.822540°E$).

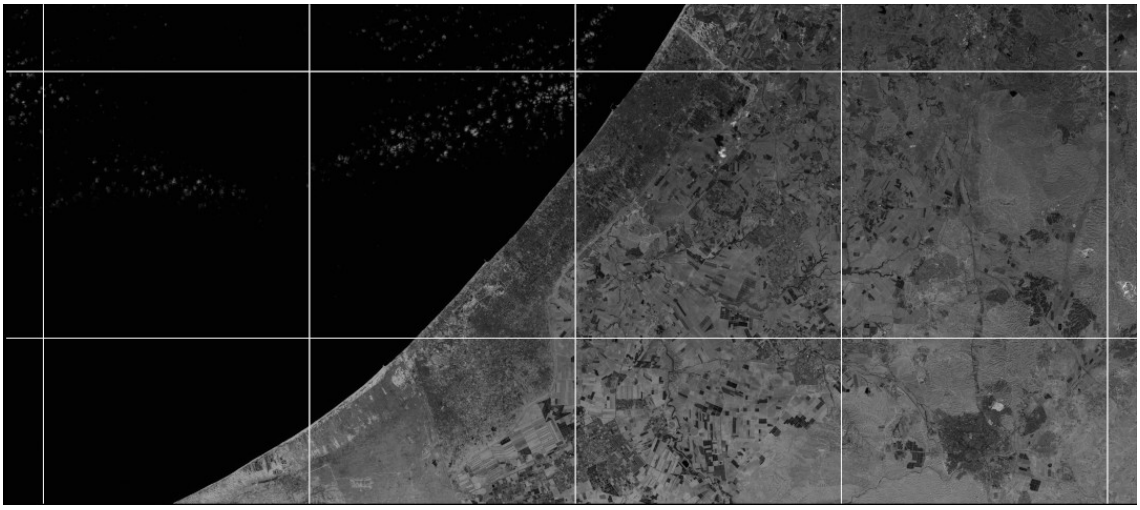


Figure 2: Our area of interest (the Gaza strip) subdivided into maplets that would not exceed the 2500×2500 pixel constraint.

We then analyzed data from the 7^{th} of October 2023 (image with few coulds), to test thresholding B12 and B02 values for bombing site detection and differentiating from clouds. It became apparent, that the south-west part of our initial area of interest did not have any ongoing bombings, which is consistent with the political context of Israel's aggression being focused on the nothern part of Gaza. Therefore, to avoid donloading unnecessary data, we reduced the scope of our test case study to a smaller range of coordinates ($31.405181° - 31.570155°N$, $34.475647° - 34.735817°E$), which is presented, together with thresholded B12 and B02 band values, on Figure 3. A threshold for values in both bands is a parameter in our code, for the test case study we used 0.75 of the maximum value (255) for both. The problem with the thresholding is that it selects individual pixels as dangerous, leading to a scattered representation (as on Figure 3) that would not be the most applicable danger heat map for our purposes. We will address this problem in section 4.

4 Danger heat map generation based on remote sensing

Our vision of the final product involves a danger heat map that would rely on a combination of different information: analysis of the sentinel bands, machine learning applied to satellite images, dangers reported by people on-site, data from the past and from other danger assessment sources such as in Section 6. In this section, however, we will show a practical example of how one of those components could be generated based on the high B12 values identified in section 3.

The concept is fairly simple: taking the pixels with high B12 and B02 values identified by the code, we will run a clustering algorithm to identify groups of pixels that would belong to the same bombing site, or the same cloud, respectively. We then associate a gaussian with each cluster. We use the locations and standard deviations of these gaussians to identify if a given B12 group coincides with a B02 group (cloud) or not. Having filtered out such false positives, we can create a heatmap by adding values of those gaussians.

For clustering the data we use a DBSCAN algorithm. This algorithm selects points with at least n_p neighbours

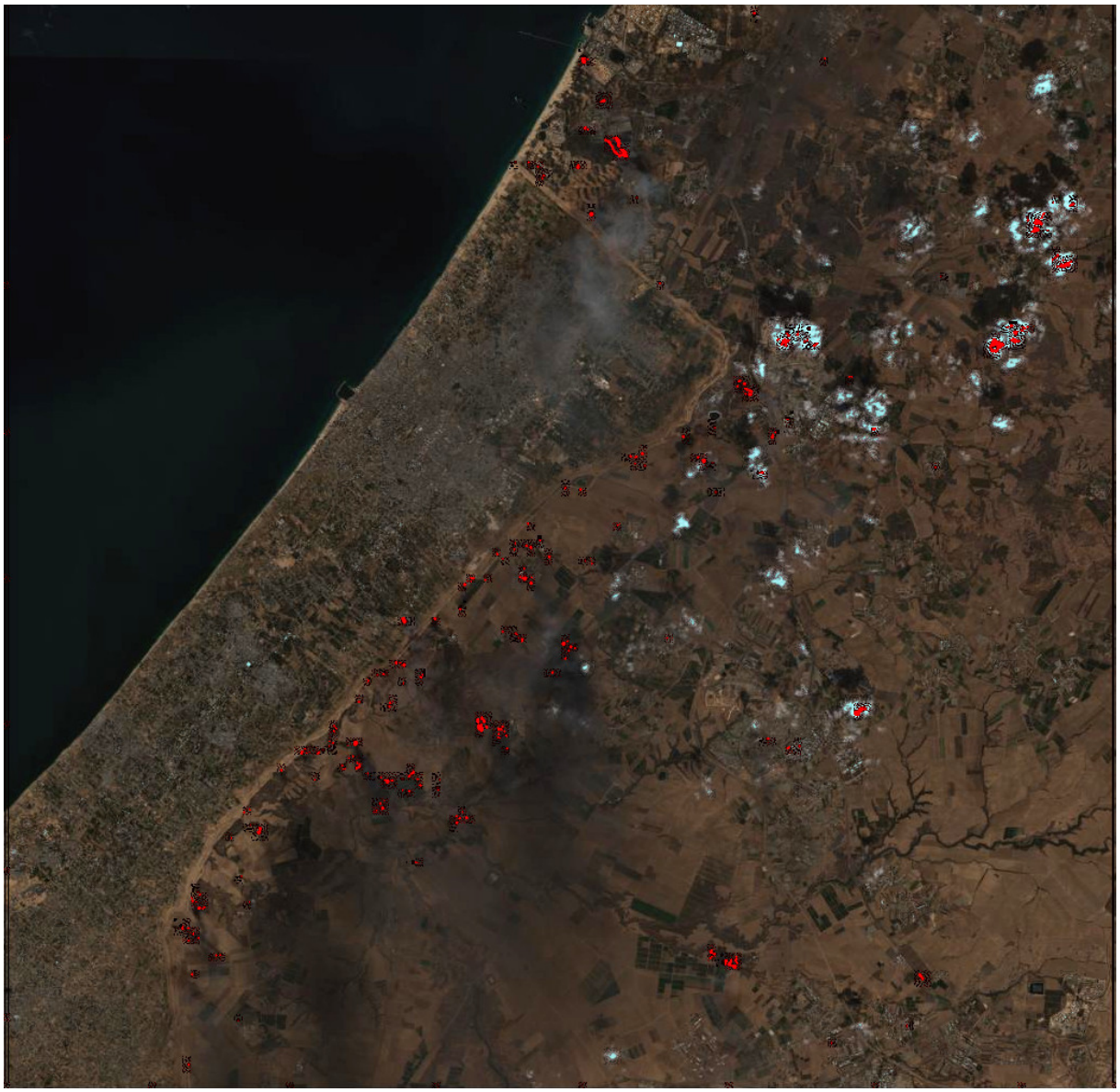


Figure 3: Bombing site identification with the Python code. Analogously as with *QGIS* in Figure 1, we select out pixels with high B12 values (red). Some of those (primarily in the central and left part of the image) correspond to actual bombing sites. Others (upper right) are clouds - false positives due to the fact that clouds have high reflectance over the entire spectrum (and hence, in every Sentinel band). We identify those by thresholding the B02 - bombing sites have very low values in that band, but for the clouds it's high. This image was generated in *GIMP* by manipulating raw images outputted by the python code. The true color image presented here has enhanced brightness and contrast for better visibility. Thresholded high B02 values (clouds) are shown in light blue, with 0.5 opacity so we can see how they coincide with white louds seen on the true color image. Thresholded B12 values are shown in red, with increased contrast to make them more prominent. The chosen threshold was $0.75 \cdot 255$ for both bands.

within an r radius as candidates for cluster cores. It then keeps adding points to the same cluster if they are found within an r radius of any point from a cluster. The n_p and r are parameters of the algorithm, here we use $n_p = 5$ and $r = 10$ (pixels).

Our final function for the heatmap is:

$$H(x,y) = \min\left(\sum_{c\in C,\ \min\limits_{cloud\in L}(d_{cloud}(\mu_c))>\gamma} \left(\log\left(1+N_c\right)\cdot e^{-d_c((x,y))^2}\right) + \alpha(\Delta t)\cdot H'(x,y), H_{\max}\right) \tag{1}$$

where

$$d_c(P=(x,y)) = \sqrt{\left(\frac{x-\mu_{cx}}{\sigma_{cx}}\right)^2 + \left(\frac{y-\mu_{cy}}{\sigma_{cy}}\right)^2} \tag{2}$$

and we will explain all the variables of the equation below:

- Equation 4 describes a distance of a single point (x, y) from a cluster c, in units normalized by the standard deviation of the cluster. μ_{cx} and $\mu_c y$ are the coordinates of a point μ_c, being a center of the cluster - they are obtained by taking averages of respective coordinates of all points in a cluster. σ_{cx} and $\sigma_c y$ are standard deviations of respective coordinates within a cluster. This means that we consider a different scale of the gaussians in each coordinate, which is reasonable - some clusters may be much more stretched out in one direction than the other, so they would not be well represented by a radially symmetrical gaussian.

- For a given cluster of $c \in C$, where C is a set of all high B12 clusters, which contains N_c points, the term $\log\left(1+N_c\right)\cdot e^{-d_c((x,y))^2}$ of equation 4 is a gaussian we will associate with that cluster. The amplitude of the gaussian equal to $\log\left(1+N_c\right)$ was chosen by experimentation - an amplitude that would be linearly proportional to N_c lead to terrible results (heat maps with very disproportionate values, impossible to be attributed with any meaningful interpretation). Notice that this is still starting out as linear for the smallest clusters ($\log(1+\xi) \approx \xi$ for small ξ).

- The summation in equation 4 goes over all clusters in C which satisfy a condition $\min\limits_{cloud\in L}(d_{cloud}(\mu_c)) > \gamma$, where L is a set of all clusters of points with high B02 values. This is essentially saying that the center of a detected bombing site should be further than γ (in a metric defined by equation 4) from any cloud (notice that we calculate the distance in units of the standard deviation of the cloud, not the B12 cluster) in order not to be considered a false positive and excluded from the summation. For our test case study, our experimentation lead us to choosing $\gamma = 1.15$.

- We add the term $\alpha(\Delta t)\cdot H'(x,y)$ to our sum of gaussians, where $H'(x,y)$ is a previous heatmap value and $\alpha(\Delta t)$ is a decay parameter dependent on Δt - time elapsed since the heatmap was last updated. This is to model a realistic scenario where we would be monitoring a site regularly and updating an existent heat map. War and occupation related danger is largely additive - if an area has been bombed or surrounded by an enemy a week ago, the danger would still be there, even if we are not detecting any activity. However, over longer time periods, if no new activity is detected, it is very plausible that the war is being waged in other locations and the former danger site is becoming safer. Therefore, we multiply the old heatmap by $\alpha(\Delta t) < 1$, which is an decresing function of Δt. For our test case study, we used $\alpha(\Delta t) = 1 - 0.01\cdot t$, where Δt is expressed in days (and for the satellite images available for Gaza in October 2023 the typical value was $\Delta t = 5 \implies \alpha(\Delta t) = 0.95$).

- Finally, an upper bound of $H_{\max}$ is introduced, so that the heat map values would not stretch onto infinity (if a danger is very high, it should just correspond to a fixed maximum value, with no further differentiation). Presumably, for the final version of the algorithm (5) we would be mapping the range of $H(x,y)$ onto a $[0,1]$ interval, or perhaps just onto a set of discrete values in that range. For now, for the sake of displaying heatmaps as images, we set $H_{\max} = 255$.

We ran a loop over all dates in October 2023 to download available data and generate heat maps in accordance to the model described (Equation 4). The result for three chosen dates (those with fewer clouds, for better representation) are shown in Figure 4.

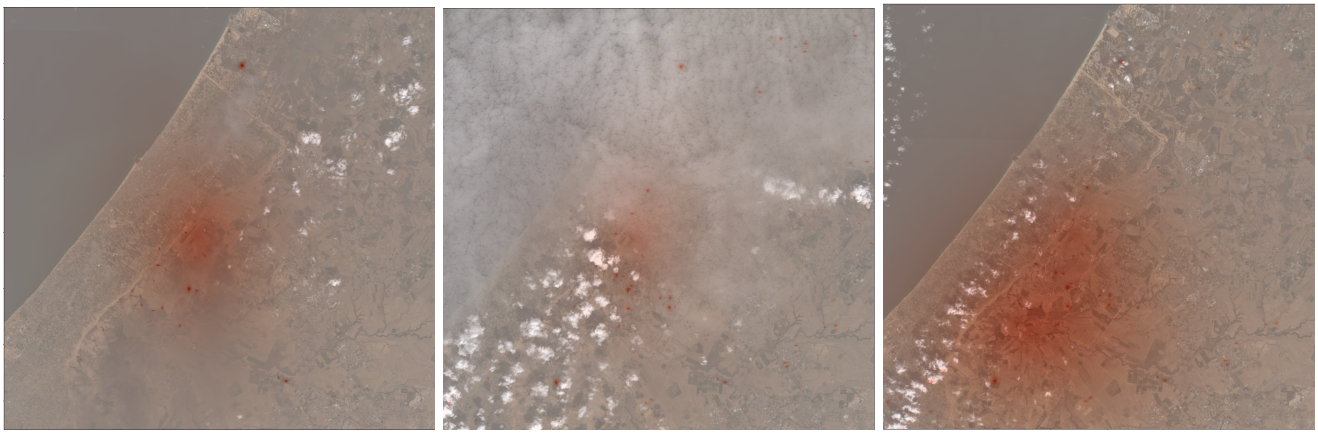
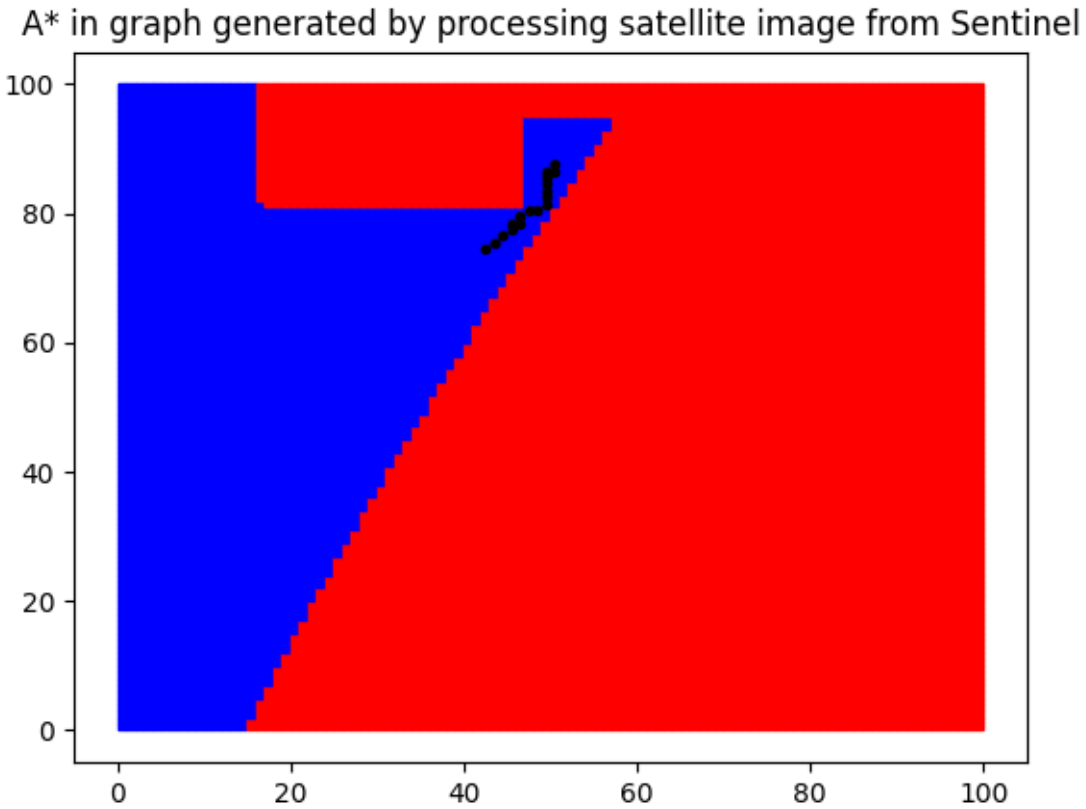


Figure 4: Danger heat maps for three different dates, as generated by our model. Dates, from left to right, are 7^{th}, 17^{th} and 22^{th} of October. The additive nature of the model time evolution is visible.

5 Path finding algorithm

We wrote A* algorithm to look for a possible path to leave areas that are in danger.

Also using closest facility and ARCGIS pro API's we want to provide function fiding closes shelter etc.



6 Research into other available data, such as UNOSAT reports

In order to get sample data for the damage assessment algorithms data. The UN's UNOSAT database (Figure 5) was employed to extract destruction points and assign arbitrary values for each class of destruction (Figure 6).

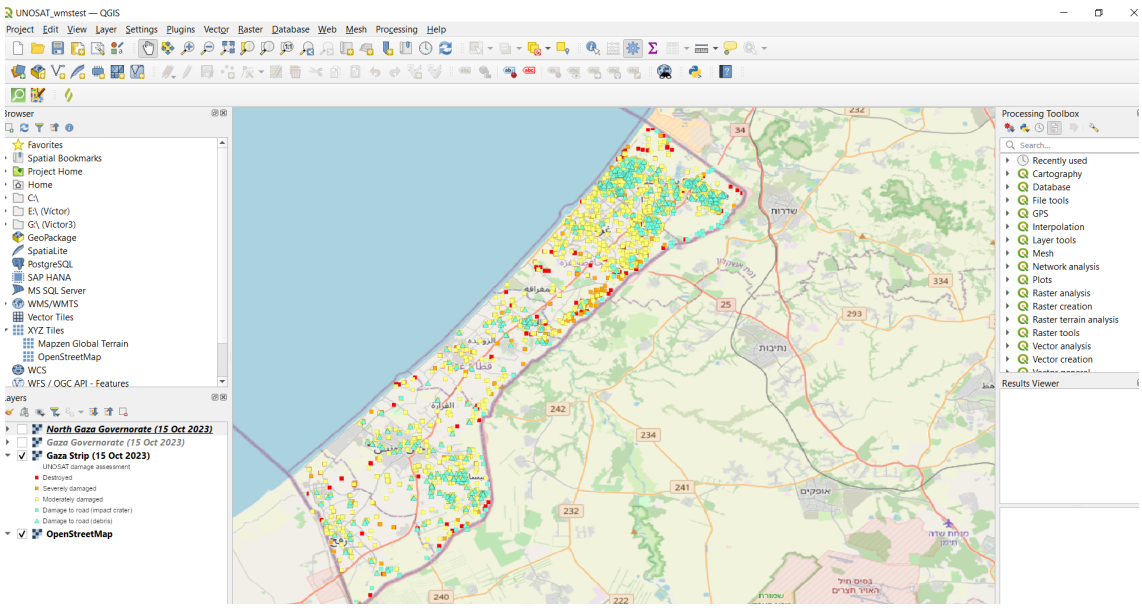


Figure 5: Damaged buildings as reported by UNOSAT displayed over a street map of the Gaza strip.

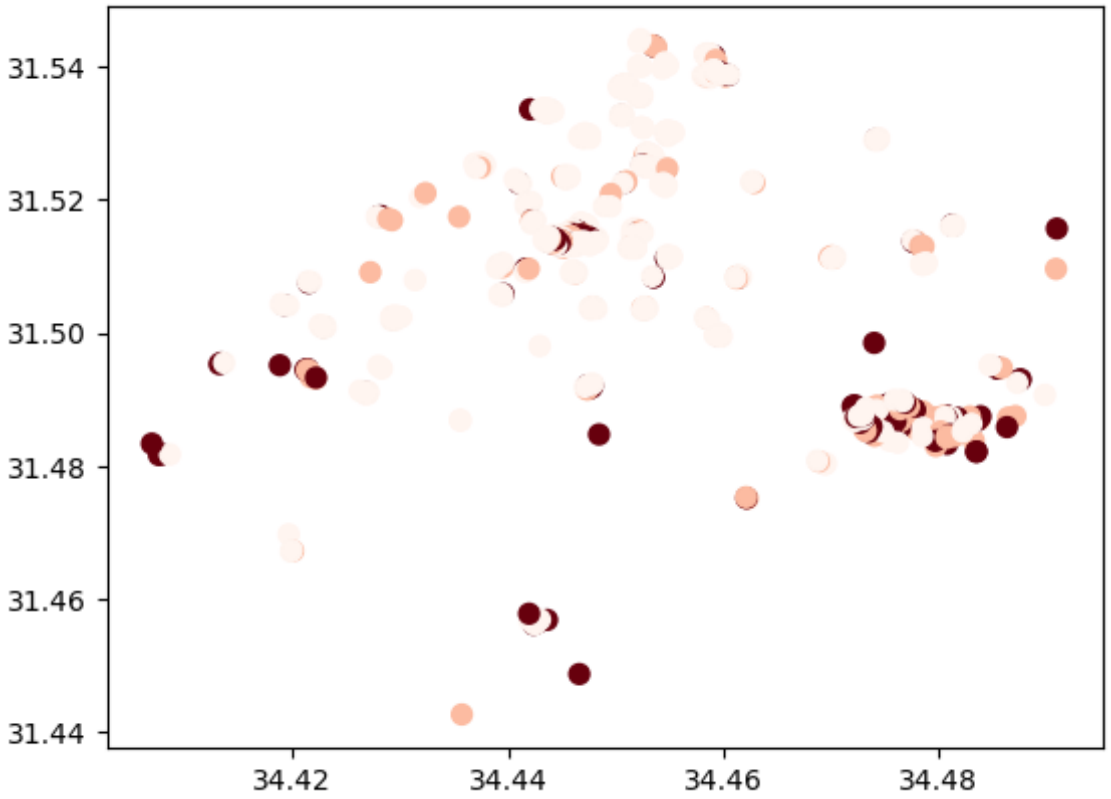


Figure 6: Reported building destruction displayed as a Python scatter plot where the darker the red tone, the higher is the destruction reported.

Once the values were assigned, a Python code was written and used to interpolate values between the located points in a sample area. Nearest neighbour, linear and cubic algorithms were employed with the linear one showing the best results (7).

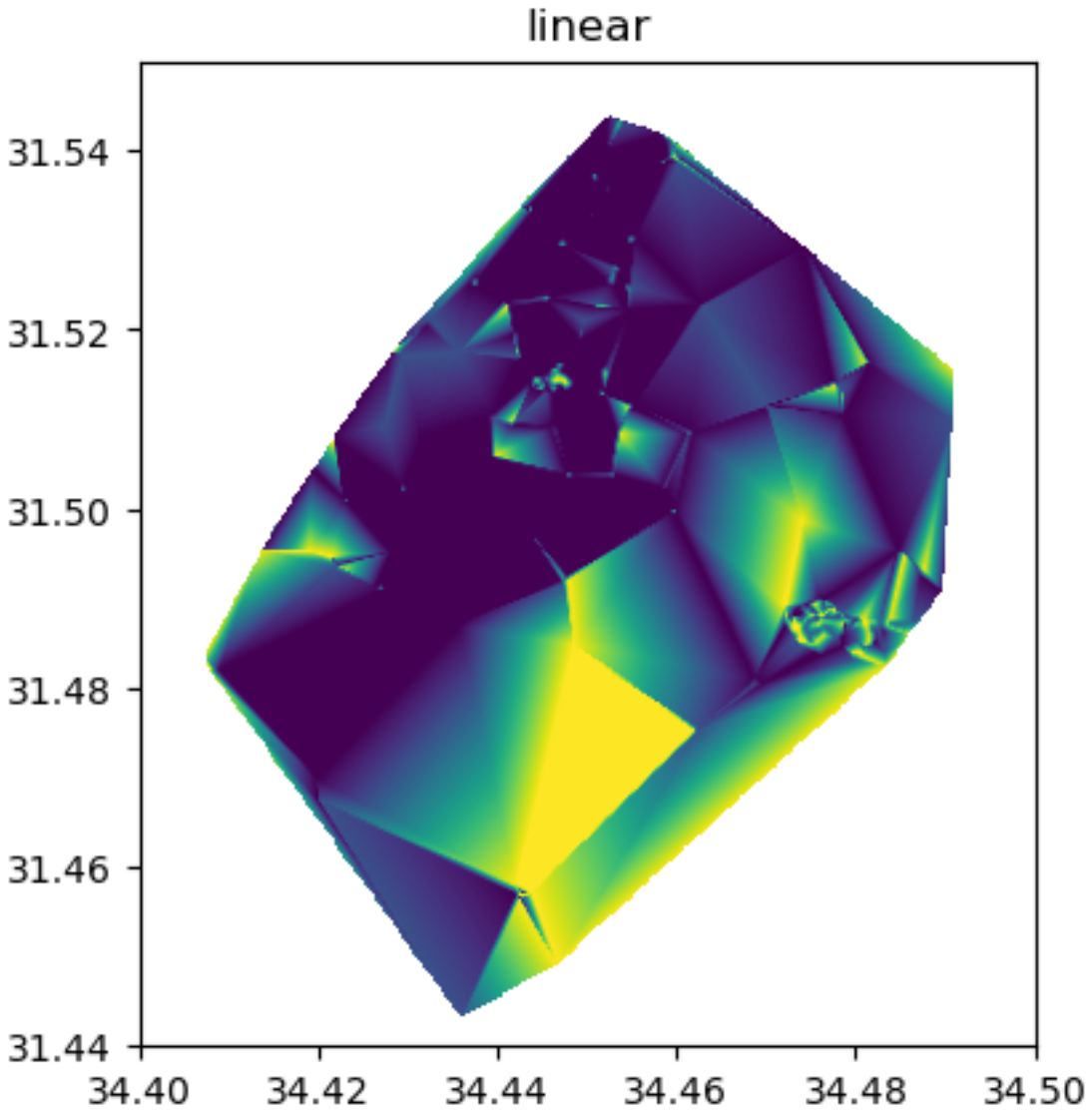


Figure 7: Linear interpolation of assigned destruction values, Higher values represent yellow areas and Blue areas represent lower values.

7 Website prototype

We have created a website available for users in endangered areas and humanitarian aid providers 8.

The site contains constantly updated maps showing civilian risk areas, ranked from the most dangerous hotspots to less threatened areas. fig9

Users have the possibility to manually input spotted danger in their area, from soldiers to tanks, shooting, and damage. All the data will be verified and can affect the percentage of risk in the area. fig10

8 Mobile App prototype

For users in hotspots and humanitarian agencies, the safest escape route is shown on the map. The website uses the geolocation of users for better navigation and safety purposes. fig11

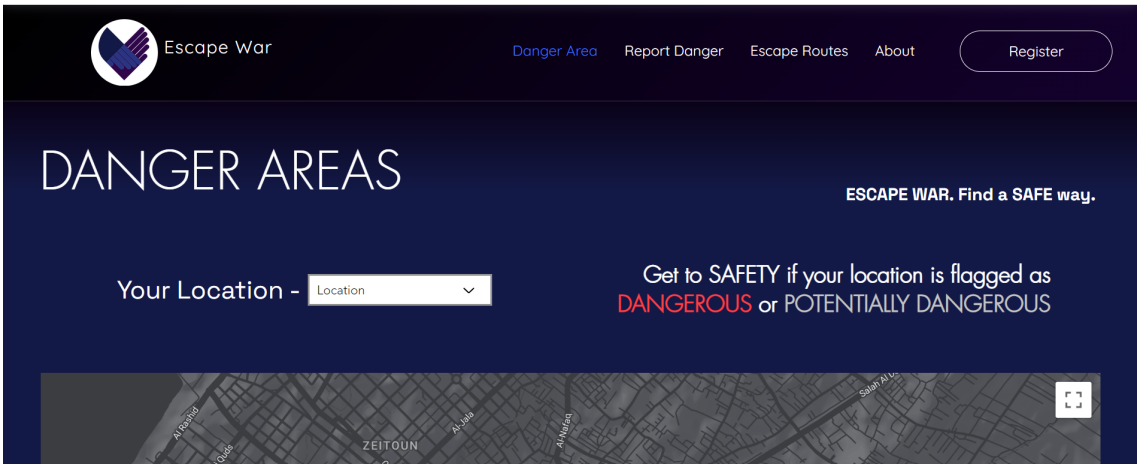


Figure 8: Main page of the website

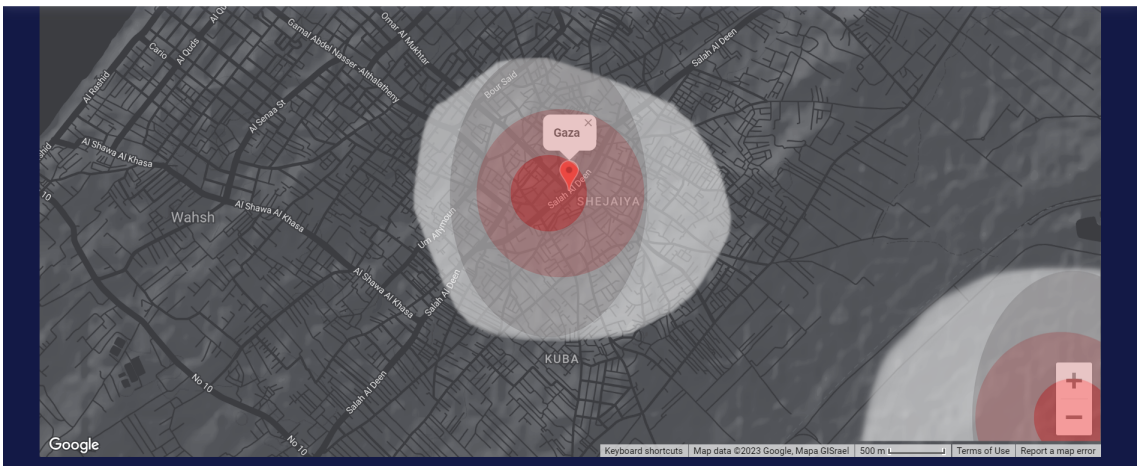


Figure 9: The map with danger zones

Figure 10: Various user input

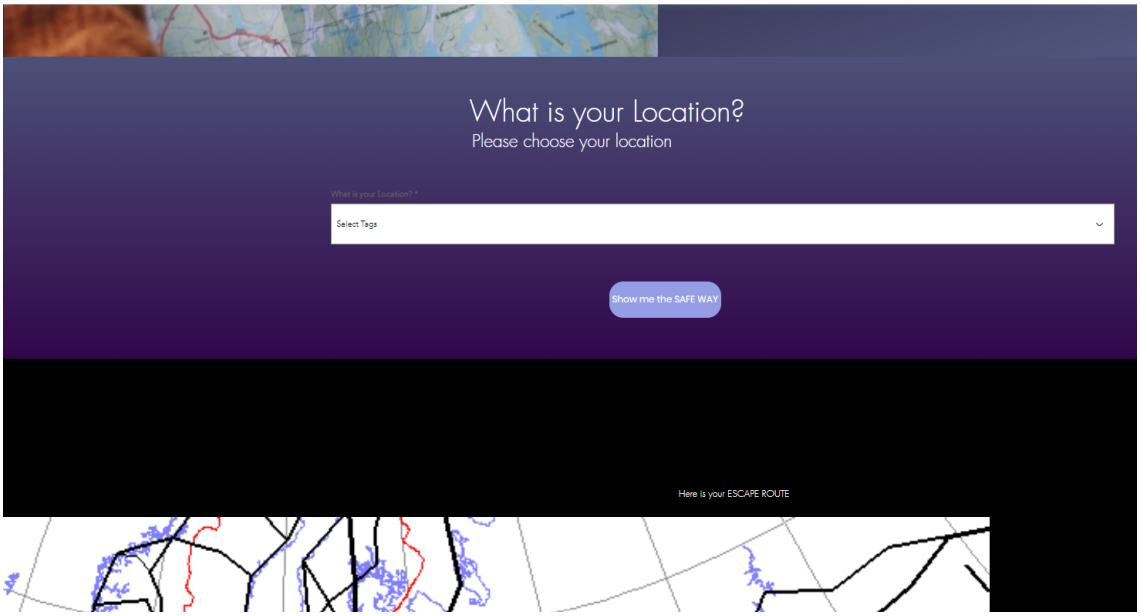


Figure 11: Safest escape route navigation

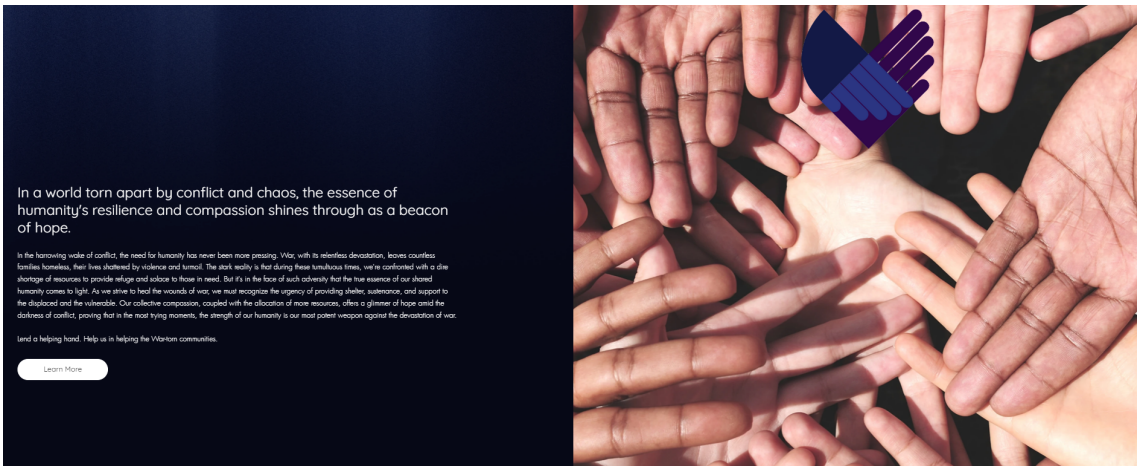


Figure 12: About us

For the purposes of advertising, finding new partners and additional funding, we would like users and customers to learn more about us. fig12

A mobile app prototype was produced as most users would be in a position were connecting through a Mobile device will be a possibility that would not depend on reaching safe zones as shelters where information will be delivered as well.