CHARLES UNIVERSITY

Faculty of Science

Study Programme: Applied Geography

Branch of Study: Social Geography and Geoinformatics

Jáchym Černík

DETECTING CHANGES IN WAR-DAMAGED URBAN AREAS USING THE IR-MAD METHOD AND SENTINEL-2 SATELLITE DATA

DETEKCE ZMĚN VE VÁLKOU POŠKOZENÝCH MĚSTSKÝCH OBLASTECH POMOCÍ METODY IR-MAD A SATELITNÍCH DAT SENTINEL-2

Supervisor: RNDr. Josef Laštovička, Ph.D.

Advisor: Mgr. Jan Svoboda

CHARLES UNIVERSITY

Faculty of Science

Department of Applied Geoinformatics and Cartography

ASSIGNMENT OF BACHELOR'S THESIS

Name and surname: Jáchym Černík

Study discipline: Social Geography and Geoinformatics

Thesis topic/title: Detecting Changes in War-Damaged Urban Areas Using the IR-MAD Method and Sentinel-2 Satellite Data

Annotation

The bachelor thesis will deal with the detection of phenomena associated with war conflicts - e.g. fires, bombing sites, etc. For the work itself, custom scripts will be programmed (in Python) using the Google Earth Engine API. Freely available Sentinel-2 multispectral imagery will be used for analysis. Change Detection method IR-MAD (Iteratively Reweighted Multivariate Alteration Detection) will be tested for a selected location of an urban area with an ongoing or past conflict to detect and quantify the changes taking place. The results will be validated using Planet's very high spatial resolution optical satellite data. A custom web-based mapping application will be developed to present the results. The application will be published on its own website, which will be created using Google Sites technology.

Objectives

1. To determine the utility of IR-MAD for detecting changes associated with war-related conflicts. Planet data with very high spatial resolution will be used to validate the accuracy of change detection.
2. Building scripts to observe, detect and validate phenomena associated with war conflicts.
3. Building a custom online web map application showing detected changes in the selected area and creating a website using Google Sites technology to present the created online map application.

Bibliography:

Aimaiti, Y., Sanon, C., Koch, M., Baise, L. G., & Moaveni, B. (2022). War Related Building Damage Assessment in Kyiv, Ukraine, Using Sentinel-1 Radar and Sentinel-2 Optical Images. In Remote Sensing (Vol. 14, Issue 24, p. 6239). MDPI AG. https://doi.org/10.3390/rs14246239

Beygi Heidarlou, H., Banj Shafiei, A., Erfanian, M., Tayyebi, A., & Alijanpour, A. (2020). Armed conflict and land-use changes: Insights from Iraq-Iran war in Zagros forests. In Forest Policy and Economics (Vol. 118, p. 102246). Elsevier BV. https://doi.org/10.1016/j.forpol.2020.102246

Fakhri, F., & Gkanatsios, I. (2021). Integration of Sentinel-1 and Sentinel-2 data for change detection: A case study in a war conflict area of Mosul city. In Remote Sensing Applications: Society and Environment (Vol. 22, p. 100505). Elsevier BV. https://doi.org/10.1016/j.rsase.2021.100505

Kussul, N., Yailymova, H., & Drozd, S. (2022). Detection of War-Damaged Agricultural Fields of Ukraine Based on Vegetation Indices Using Sentinel-2 Data. In 2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT). 2022 12th International Conference on Dependable Systems, Services and Technologies (DESSERT). IEEE. https://doi.org/10.1109/dessert58054.2022.10018739

Marx, A. J. (2016). Detecting urban destruction in Syria: A Landsat-based approach. In Remote Sensing Applications: Society and Environment (Vol. 4, pp. 30–36). Elsevier BV. https://doi.org/10.1016/j.rsase.2016.04.005

Supervisor of bachelor's thesis: RNDr. Laštovička Josef, Ph.D.

Date of assignment of bachelor's thesis: 27. 6. 2024

Bachelor's thesis submission deadline: September 2024 Prague, on 11. 7. 2024

Supervisor's signature

Declaration

I hereby affirm that the entirety of this thesis is my own work, with all sources appropriately cited. I attest that this thesis, in whole or in part, has not been submitted for any other academic degree at any institution.

In Prague, July 2024
Jáchym Černík

Acknowledgement

I would like to express my deepest gratitude to my thesis supervisor, RNDr. Josef Laštovička, Ph.D., for his invaluable guidance, unwavering support, and constructive feedback throughout the entire research process. His expertise and counsel made the completion of this work possible. Additionally, I am sincerely thankful to Mgr. Jan Svoboda for his insightful guidance in understanding Multivariate Alteration Detection and his valuable feedback. Lastly, I extend my thanks to Dr. Mort Canty for providing accessible and exceptional tutorials that make research endeavors like this possible.

Abstract

This study focuses on Sentinel-2 multispectral data for detecting changes associated with war conflicts in an urban environment in Google Earth Engine (GEE) cloud based platform. The area of interest was chosen to be the Gaza City and its surrounding, which became embroiled in a military conflict in October 2023. The Python scripts have been used to perform analyses and monitor spectral signs over time. The iteratively reweighted multivariate alteration detection (IR-MAD) method, which is based on the comparison of two images, was used to analyze the changes. The resulting raster of changes was validated with very high spatial resolution PlanetScope data. Based on the validation, an overall accuracy of 74% was achieved. As part of the research, a web-based mapping application was created to allow users to view conflict using pre-built tools.

Key words: Change Detection, IR-MAD, Google Earth Engine, Gaza,

Abstrakt

Tato studie se zaměřuje na multispektrální data Sentinel-2 pro detekci změn spojených s válečnými konflikty v městském prostředí v cloudové platformě Google Earth Engine (GEE). Jako zájmová oblast bylo vybráno město Gaza a blízké okolí, které se v říjnu 2023 zapletlo do válečného konfliktu. K provádění analýz a sledování spektrálních příznaků v čase byly použity skripty v jazyce Python. K analýze změn byla použita metoda iterativně převážené vícerozměrné detekce změn (IR-MAD), která je založena na porovnání dvou snímků. Výsledný rastr změn byl ověřen pomocí dat PlanetScope s velmi vysokým prostorovým rozlišením. Na základě validace bylo dosaženo celkové přesnosti 74 %. V rámci výzkumu byla vytvořena webová mapová aplikace, která umožňuje uživatelům prohlížet konflikty pomocí předem připravených nástrojů.

Klíčová slova: Detekce změn, IR-MAD, Google Earth Engine, Gaza,

Table of Contents

Abbreviations

AOI Area of Interest

API Application programming interface

EEA European Environment agency

CCA Canonical Correlation Analysis

CD Change Detection

CV Canonical Variates

CVA Change Vector Analysis

GEE Google Earth Engine

GFW Global Forest Watch

GLAD Global Land Analysis and Discovery

IDE Integrated Development Enviroment

IR-MAD Iteratively reweighted Multivariate Alteration Detection

LCMAP Land Change Monitoring Assessment, and projection

LUCC Land use and Land cover change

MAD Multivariate Alteration Detection

MSI Multispectral Instrument

PCA Principal Component Analysis

RS Remote Sensing

SAR Synthetic Aperture Radar

TSA Time Series Analysis

UN United Nations

UNITAR United Nations Institute for Training and Research

UNOSAT United Nations Satellite Center

USGS The United States Geological Survey

VHR Very High Resolution

List of Figures

List of Tables

1. Introduction

Remote sensing (RS) is commonly utilized to assess changes attributed to natural disasters such as floods and earthquakes. Its potential to observe anthropogenic activities makes it a valuable tool for monitoring urban destruction from human conflicts and wars. Collateral damage is a common phenomenon in urban areas affected by modern warfare, yet there is a lack of a reliable framework for reporting it. While professional organizations like the United Nations Satellite Centre (UNOSAT) provide organized reporting, human monitoring, though accurate, can be limited in speed. Additionally, it can also come with higher costs. The need for an automated method that can monitor damage objectively and without bias is crucial, especially in an internet era where public information can be skewed.

Consequently, the primary objective of this research is to employ a comparatively robust Iteratively Reweighted Multivariate Alteration Detection (IR-MAD) technique developed by Nielson (1998). The selection of this method was informed by its prior successful application in detecting changes in appliances such as monitoring forest cover change (La Barreda-Bautista, A., Couturier, Luis 2011), in other cases IR-MAD was combined with object detection to discover illegal rooftop construction (Liu et al. 2021). This only proves the versatility of this method. Therefore, the versatility of this method allows for an opportunity to observe urban changes attributed to warfare. However, urban environments exhibit significantly less stability for RS, necessitating preprocessing of imagery to mitigate the potential for false changes arising from diverse reflectances (Herold, Roberts 2010).

The initial task involves the implementation of the IR-MAD code provided by Canty, which is accessible through the "Change Detection in Google Earth Engine - The MAD Transformation tutorial" (Canty 2024). The code is applied to a selected area of interest (AOI) during two time periods: one before a conflict and the other after the conflict. The results are then evaluated against reference data. Tweaking and masking are incorporated into the process. The primary objective of this task is to determine the suitability and accuracy of IR-MAD as a method for monitoring changes in urban areas affected by warfare. In the second task, the Sentinel-2 data will be statistically evaluated with PlanetoScope data with very high spatial resolution to determine the accuracy of IR-MAD results.

The main objectives of this study were:

- To utilize and test IR-MAD with Sentinel-2 data in the use of CD in urban areas affected by the war. The CD will be validated with very high spatial resolution PlanetScope data.
- To create Python scripts for CD analyses.
- To develop a website with a web mapping application for publishing the results.

2. Change Detection

This chapter aims to provide a comprehensive review of relevant publications on CD in RS. It is structured into three subsections. The first subsection focuses on the characteristics of CD discipline in RS. The second subsection examines CD methods employed in monitoring war-related damage. Lastly, the third subsection explores some of the applications of IR-MAD specifically for CD.

CD is the process of identifying differences in the state of an object or phenomenon by observing it at different times (Singh 1989). Timely and accurate CD of the earth's surface is crucial for better understanding relationships and interactions between human and natural phenomena (Lu, Mausel, Brondízio, Moran 2004). This allows us to identify trends, monitor processes, and gain a deeper understanding of Earth's dynamics.

The fundamental principle of CD is that changes in the earth's surface should result in corresponding changes in digital number values. However, to be considered significant, these surface changes must be more substantial than radiance variations caused by other factors like the sun's angle, atmospheric conditions, or variations in surface moisture (Singh 1989; Jensen 1983).

2.1. Procedure

According to Lu et al (2004) a CD research should provide the amount of area changed and change rate, spatial distribution of changed types, trajectories of land-cover types and accuracy assessment of CD results. These characteristics are optimal, it is however up to many factors that affect the result of CD analysis. These factors can be seen in Table 1.

Table 1 : Factors affecting the results of CD

Category	Factor
Image Preprocessing	(1) Precise geometric registration (2) Calibration or normalization
Ground Truth	(3) Availability of quality ground truth data
Study Area Characteristics	(4) Complexity of landscape and environment
Methodology	(5) CD methods/algorithms (6) Classification and CD schemes
Analyst Expertise	(7) Analyst's skills and experience (8) Knowledge/familiarity of study area
Resource Constraints	(9) Time and cost restriction

Source: (Lu, Mausel, Brondízio, Moran 2004)

Considering these factors, it is important to note that for CD to be feasible in RS, precise geometric registration and calibration are desiired. Therefore, multitemporal images need to be preprocessed to ensure they are radiometrically and spatially comparable. Geometric correction ensures that pixels in multitemporal images refer to the same geographic location. Normalization and calibration address differences in imaging seasons, solar angles, and meteorological conditions. Once preprocessing is complete, the multitemporal data should be ready for CD analysis. (Ban, Yousif 2016)

Another step in the CD framework is the characterization of change information, including the types of CD products, their characteristics, and methods, as depicted in Table 2. While numerous approaches exist to depict change, two methods are particularly prevalent. First, the binary change map synthesizes bitemporal CD data to highlight the spatial location of changes. Second, the "From-To" change map, a typical product of post-classification CD, emphasizes transitions between different land cover types by displaying the differences between assigned values, “From-To” map is exemplified in the study by El-Hattab (2016) “*Applying post-classification CD technique to monitor an Egyptian coastal zone* “ where change information has been displayed through negative, positive and no change, for each class by comparing base-classification to post-classification.

Other frequently used products in Remote Sensing (RS) include those depicting the magnitude and direction of change, which are essential outputs through Change Vector Analysis (CVA). In CVA, change vectors are calculated by subtracting vectors on a pixel-by-pixel basis, similar to image differencing. The magnitude and direction of these change vectors can be visualized into change information. Specifically, the magnitude represents the significance or intensity of the change, while the direction indicates the nature or type of change.

An example of this application can be found in the study by Rahman and Mesev (2019), titled “*Change Vector Analysis, Tasseled Cap, and NDVI-NDMI for Measuring Land Use/Cover Changes Caused by Short-Term Severe Drought: The 2011 Texas Event*.” In this research, the authors utilized CVA on NDVI-NDMI to measure the magnitude of changes, producing a continuous scale that highlighted both significant and insignificant changes. The direction of change was also depicted for NDVI-NDMI, revealing changes towards increased wetness, alterations in chlorophyll content, bare-soil expansion, and decreased wetness. The magnitude of change is also an output of the Land Change Monitoring, Assessment, and Projection (LCMAP) program, which will be elaborated upon later in this chapter.

Another notable approach involves identifying the likelihood of change occurring. This product is an estimated value representing the probability of change in a specific area, often derived from temporal analysis of historical datasets. By calculating this probability, it becomes possible to predict future conditions, which can aid in prevention and planning research

(Panuju, Paull, Griffin 2020). Estimating change probability has been implemented in various applications, for example Chen et al. (2017) used a probabilistic approach to determine the likelihoods of landslide-formed lakes forming. In another study, an estimation model was utilized to measure the susceptibility of areas to floods (Samanta, Pal, Palsamanta 2018). Similarly, Liang & Liu (2020) used a probabilistic model to estimate daily changes of inundation related to storm surges and river floods.

Temporal change trajectories, or Land Use and Cover (LUCC) trajectories, are periodically observed sequences of land cover changes in pixels or objects, integrated into serial datasets. This category includes both temporal trends and seasonal patterns. Temporal trends focus on long-term directional changes, typically derived from aggregated data from periodic or non-periodic observations. Seasonal patterns encompass seasonal, trend, and irregular patterns in terms of fluctuations in regular and consistent intervals (Panuju, Paull, Griffin 2020). However, both of them utilize the same techniques such as Time Series Analysis (TSA), for examining data collected or recorded at successive points in time to identify the pattern or trend. For example, Huesca et al. (2015) used MODIS based TSA for monitoring seasonal patterns to assess changes in ecosystems through seasons. On the other hand, temporal trend has been observed in Biomass estimations. Browning et al. (2017) aimed at distinguishing seasonal patterns from long term trends to discover changes in the amount of biomass using TSA with NDVI.

Lastly, Dynamic Simulation of Changes is a CD product used to simulate future land cover changes. It often utilizes heterogeneous data processed by Geographic Information Systems (GIS) and spatial analysis. This approach aims to interpret future changes by employing simulation models to predict land cover transformations. This category encompasses both anthropogenic and natural phenomena. For instance, Getu & Bhat (2022) utilized Cellular Automata and Markov models, combined with urban growth driver data and past TSA satellite imagery, to study and simulate future land use changes in the urban area of Bahir Dar city, Ethiopia. These methods often employ heterogeneous data to provide more inputs to the model.

Table 2 : CD Products, their characteristics and methods.

CD products	**Characteristic**	**Method(s)**	**Examples**
Binary Change	Identifies whether change has occurred or not	Image differencing, thresholding, CVA, MAD, IR-MAD, Slow Feature Analysis (SFA), deep learning	Urban (Doxani, Karantzalos, Strati 2012) Buildings (Peng, Guan 2019) Forest damage (Hamdi, Brandmeier, Straub 2019)
Types of Change: "From-To" Information	Determines the specific nature of change	Post-classification comparison, multi-temporal classification, object-based CD	Land Cover change (El-Hattab 2016) Agriculture (Wang, Guanzhou, Dai, Gong, Zhu 2018)
Magnitude and Direction of Change	Quantifies the amount and direction of change	Change Vector Analysis	Drought assessment (Rahman, Mesev 2019)
The Probability of Change	Estimates the likelihood of change occurring	Chi-square distribution, likelihood ratio, Dempster-Shafer theory	Flood susceptibility (Samanta, Pal, Palsamanta 2018) Landslide lakes (Chen, Li, Zhang, Jiang, Tao, Shen 2017)
Temporal Change Trajectories. Temporal Trend Seasonal Pattern	Tracks changes in individual locations over time	Time series analysis,Seasonal trend analysis,curve fitting,	Ecosystem assessment (Huesca et al. 2015) Abandoned agriculture.(Alcantara, Kuemmerle, Prishchepov, Radeloff 2012) Biomass estimation (Browning, Maynard, Karl, Peters 2017)
Dynamic Simulation of Changes	Models of change processes over time	Land-use change models, cellular automata, Markov chain models,	Landuse change (Islam, Rahman, Jashimuddin 2018) Urban expansion (Getu, Bhat 2022)

Source: (Panuju, Paull, Griffin 2020)

Choosing the right method is as important as choosing the right data. Researchers need to consider their objective and which method is best. Cheng et al. (2023) summarized CD methods based on their algorithm granularity into three groups:

- **Pixel-based methods** focus on the individual pixel as the basic unit of analysis. In these methods, changes are detected and measured by analyzing the spectral characteristics of each pixel, largely disregarding the spatial context. This approach treats pixels as independent entities, if changes in the Earth's surface correspond directly to changes in the pixel's spectral values. However, this assumption can be limiting, as it overlooks the spatial relationships between pixels and the broader context of the image. On the other side, Pixel-based are often the backbone of other methods, additionally they are effective due to their simplicity.
- **Object(region) based methods,** which take spatial context into consideration, address the limitations of pixel-based approaches by grouping pixels in an image to form image objects that correspond to meaningful entities in the scene. This method relies on image segmentation and often employs various classification techniques to accurately delineate and categorize these objects.
- **Hybrid CD methods** leverage the strengths of two or more individual techniques, drawing from both pixel-based and object-based approaches. By combining methodologies, hybrid approaches can overcome the limitations inherent in each method when used in isolation. For instance, they might utilize the spectral precision of pixel-based methods while incorporating the spatial context provided by object-based techniques.

Pixel-based techniques were pioneering CD in the early days of RS, as the processing power required for object-based methods emerged later. Generally, increased granularity correlates with method complexity. While object-oriented methods offer potential for higher accuracy, they introduce additional steps to the CD process, such as image segmentation. If not executed optimally, segmentation can skew results in object-based methods. Additionally, pixel-based methods are often suboptimal for CD in Very High Resolution (VHR) imagery due to increased spectral variability, and their limitations of comparing atmospheric noise. Object-oriented and hybrid approaches can address this issue by incorporating classification or other contextual data, overcoming the limitations of simpler techniques in VHR scenarios. (Hussain et. 2013). Hybrid CD methods, while potentially yielding the most accurate and insightful results, often demand a high degree of optimization. This is evident in " *Detecting land use changes using hybrid machine learning methods in the Australian tropical regions"* Sedighkia & Datta (2023) While the study used hybrid Machine learning methods together against conventional and robust methods for comparison, the authors found that conventional methods were more robust and less time consuming due to high computational complexities. Researchers must therefore balance the choice between straightforward methods with potential limitations and complex methods that may offer better results but require significant optimization and computational power

Alternatively CD methods can be distinguished by their change information acquisition process. Asokan & Anitha (2019) summarizes a taxonomy with following categories.

- **Algebra based CD** involves applying raster oriented algebraic operations to each image pixel to calculate the difference between images. This category includes traditional methods like image differencing, image regression, and Change Vector Analysis (CVA). These methods share characteristics such as relative simplicity (with the exception of CVA) and the need to establish thresholds for identifying changed areas. Finding the thresholds may prove to be difficult. (Asokan, Anitha 2019; Lu, Mausel, Brondízio, Moran 2004)
- **Transformation based CD** involves methods that transform image pixels to find change information from multi-temporal data. This category includes techniques such as Principal Component Analysis (PCA), MAD, and Tasseled Cap transformation. These methods usually reduce data redundancy between bands and generate new component / band or components / bands containing derived information. However, each analysis is scene-dependent and results can sometimes be difficult to interpret. Thresholding is required to identify change areas within these new components. (Asokan, Anitha 2019; Lu, Mausel, Brondízio, Moran 2004)

- **Classification based CD** is made out of different classification techniques such as unsupervised CD. The main advantage of this method is that it can provide accurate change information which is not much affected by external factors, on the other hand classification creates classes which may later hide changes within classes that could be interesting. (Asokan, Anitha 2019) Additionally, subsequent classification and comparison is constrained to the initial set of class labels. (Deer 1995).
- **Advanced models based CD** methods use different reflectance and spectral mixture models. The main idea behind these methods is to convert image reflectance values into physical parameters such as vegetation information. These physical parameters are then compared between observational periods.
- **Neural network and fuzzy/deep learning methods** combine RS techniques like neural networks and fuzzy modeling (Asokan, Anitha 2019). Fuzzy Clustering as opposed to conventional clustering allows for more than one clustering class (URL 1). These methods have an advantage in allowing for the distinction of different types of changes, and therefore, different types of phenomena that are too complex for classical algorithms (algebraic or transformation-based methods). While this is certainly an advantage, pre-processing is crucial for a successful model, whether supervised, which requires an extensive amount of labeled data to train the network, or unsupervised, which still needs interpretation and computing power. (Parelius 2023)
- **Geographic information systems (GIS) methods** allow for integration of RS and GIS, in general it is a synthesis of using multispectral data with other geographic data, however CD can be solely just GIS based (Lu, Mausel, Brondízio, Moran 2004). This is a significant advantage, especially in urbanized areas where additional spatial data beyond satellite imagery is available, such as old maps, land registries, or land use records. Land registries, for example, can be used in conjunction with RS data for updating land register through CD. (Jovanović, Gavrilović, Sladić, Radulović, Govedarica 2021)
- **Visual analysis** involves visual interpretation of multitemporal image composites and on-screen digitizing of changed areas. This method was largely utilized in the early years of RS, however, it is still being used today, largely by UNOSAT in damage assessment reports. A skilled analyst can incorporate their knowledge of data and the

spatial context of the image to accurately detect change; however, it is very time-consuming. (Lu, Mausel, Brondízio, Moran 2004)

In reality, fusion of methods in research is being used encompassing all of these categories. They all have their advantages and disadvantages and specific use cases and therefore are utilized side by side, it is however, important to note that with increasing computational innovation and advances in machine learning's deep learning approaches are beginning to form a narrative for CD for future as their precision and efficiency has been increasing in past few years and is set to progress more. (Afaq, Manocha 2021)

2.2. Data for Change Detection

Choosing data for CD is synonymous with choosing the method because the selection of appropriate data is crucial for detecting and analyzing changes over time. Data plays a vital role in determining the method which should be used for CD and vise versa. There are 5 main groups of data in RS partly summarized by Cheng et al. (2023) . Their overview can be seen in Table 3.

Table 3: Overview of available data

Data	Image Attributes	Application Scenarios	Advantages	Disadvantages
SAR	Electromagnetic signals	Surface mapping	Penetrates clouds/smoke	Noisy, distortion, complex processing
Multispectral	Discrete spectral bands, visible/IR	Land cover, vegetation analysis	Versatile, easy to interpret	spectral resolution, atmospheric condition
Hyperspectral	Continuous spectral bands	Material identification, subtle change	High spectral resolution, detects change	Complex analysis, large data, spatial resolution
3D Data (LiDAR)	Point clouds, elevation	Terrain mapping, urban modeling, forestry	Precise 3D structure	Expensive, limited coverage, weather
Heterogeneous	Multi-sensor, multi-temporal	Multi-temporal analysis	Combines data strengths	Preprocessing, registration challenges

Source: Cheng et al. (2023)

Synthetic Aperture Radar (SAR) is a powerful imaging technology that utilizes electromagnetic signals to create detailed two-dimensional or three-dimensional maps of the Earth's surface. Unlike traditional cameras that rely on sunlight, SAR acts as its own emitter, making it capable of capturing images day and night, and even penetrating through clouds and vegetation (URL 2). However, SAR data does have its disadvantages. The information gathered by SAR systems is susceptible to geometric distortions, electromagnetic interference, and speckle noise. These factors can complicate the analysis of SAR images, which need to be addressed in industrial applications.

SAR is a valuable tool for various applications, including monitoring forests, tracking agricultural growth, mapping coastlines, and studying urban environments. For example, Li et al. (2019) used image differencing with Sentinel-1's SAR imagery to conduct urban building CD in Nanjing City, China. The research showed promising results after optimizing both images for a weighted difference image to combat speckle noise. Radar data can also detect building damage. Kim, Park & Lee (2023) used the Kompsat-5 to perform a texture analysis of building damage caused by the 2016 earthquake in Japan, achieving 72.5% grid-based accuracy. SAR also allows for moisture monitoring because dry soils are more permeable to radar waves, whereas moist soil tends to absorb radar energy. The difference in reflection back to the active sensor can indicate soil moisture. Dilip et al. (2023) used Sentinel-1 data to develop a drought index for monitoring recurrent early-season droughts in India triggered by delayed monsoons.

Multi-spectral data “comprises a set of co-registered images, each of which captures the spatially varying brightness of a scene in a specific spectral band, or electromagnetic wavelength region” (Warner 2017). It can be used to identify various features on the earth’s surface based on their spectral characteristics. Other benefits include inexpensiveness and accessibility. Conversely, openly available imagery has limited spectral resolution and images are affected both by interfering atmospheric conditions and seasonal changes. Due to its availability and versatility, the use-cases of multispectral data are virtually “unlimited”. One notable example involving multispectral usability is the creation of a near real-time change detection system for identifying changes using Sentinel-2 data in forests in Mexico and Colombia. This system achieved a 92.5% accuracy by incorporating classification and cloud computing for classifying classes in the near real-time (Pacheco-Pascagaza et al. 2022).

Multispectral data was an optimal choice for such a task because vegetation exhibits high reflectance in the near-infrared (NIR) region of the electromagnetic spectrum.

Hyper-spectral data whilst being similar to multispectral data capture much larger quantities of contiguous and narrow spectral bands across the electromagnetic spectrum. This can help distinguish materials with similar spectral signatures such as smooth urban surfaces and water. On the other hand, hyper-spectral sensors are relatively expensive and with limited spatial resolution and with a lot of redundant information. An application scenario for this type of data would be when there would be a need to identify specific and detailed spectral signatures. (You, Cao, Zhou 2020). Seydi & Hasanlou (2021) argued that hyper-spectral data, with its higher spectral resolution compared to multispectral data, enhances CD of similar targets. They tested this by implementing Convolutional Neural Networks and spectral unmixing to detect binary seasonal changes in farmland in China and the USA. The results demonstrated over 90% accuracy in both cases. However, despite these promising outcomes, research on hyper-spectral CD remains limited compared to the extensive studies utilizing multispectral data. Researchers still have to address the challenges posed by its disadvantages while also contending with limitations inherent to both hyperspectral and multispectral data, such as atmospheric conditions.

3D CD data are closely tied to Light Detection and Ranging (LiDAR) technology, which uses laser pulses to measure distances between the sensor and the ground or other objects. This data allows for the observation of changes in relief or object heights, particularly in urban environments. LiDAR-produced point clouds can detect changes in urban greenery, building construction, and demolition. Therefore, 3D CDs could be used for land-use monitoring, construction monitoring, and the detection of illegal construction activities (Stilla, Xu 2023). For instance, in one research example utilizing point clouds, the authors used onboard LiDAR to detect changes in traffic and seasonal vegetation by comparing LiDAR-generated point clouds with studied city's 3D model. The collected data was then processed through Markov random field based change extraction. The research successfully identified cars, pedestrians, and seasonal changes in vegetation. (Zováthi, Nagy, Benedek 2022)

Another way to work with 3D data, known as Digital Elevation Model (DEM) differencing, is used to detect changes between DEMs by subtracting one from the other to identify variations. An example of this method is illustrated in a study where researchers compared Digital Elevation Models (DEMs) created using LiDAR with probabilistic CD and support vector machines to detect landslides adjacent to roads. These results were then compared to official surveys of the terrain. The study successfully identified 53 out of the 80 landslides documented in the official survey and even detected additional landslides that had not been identified by the survey. (Mora et al. 2018)

Heterogeneous data combines information from multiple sources, such as spectral and SAR data, to provide more comprehensive and accurate CD results. By integrating these diverse data sources, the strengths of each type can be leveraged to enhance the overall analysis. However, integration of more data types can be complicated and can often lead to noise and errors in the fusion process. In case that the data share different temporal inconsistencies, it can make it difficult to compare for different information. Therefore, an application scenario would be when the sensors combined in this data have higher temporal frequency. (Stilla, Xu 2023)

The fusion of data has been utilized in numerous studies, with effective examples highlighting its importance in predictive models, where the use of more inputs may lead to better prediction accuracy. Ahmad et al. (2023) used classification on Landsat 8, supplemented with additional data such as road networks and elevation, to predict future changes using the Modules of Land Use Change Evaluation. This approach successfully estimated that the urban area of Lahore will increase by 23.15% by 2040.

In conclusion, remote sensing offers a plethora of powerful and informative data sources for CD. Each technology, from the detailed topographic data of LiDAR to the all-weather capabilities of SAR, brings its own strengths and weaknesses. Hyper-spectral data excels in detecting changes in specific objects but can be challenging to work with due to data redundancy and computing costs, while multispectral data is versatile and user-friendly but may fall short in spectral resolution or be affected by atmospheric conditions. SAR data is advantageous for its ability to operate day and night but is susceptible to noise. Similarly, 3D data allows for precise measurements and clear CD but is currently only useful on a local scale such in urban space or terrain based CDs. Lastly, heterogeneous data holds significant potential as computing power increases and machine learning advances, but practical application can be difficult due to variations in data sources and the need for extensive optimization.

2.2.1. Past and Current Datasets

Numerous CD datasets have been published and are being operated. These datasets are usually products of various institutions seeking to uncover changes using satellite data. They are not only results of previously stated methods and data, but also represent collective effort and progress in the field. These datasets differ from other RS datasets such as Corine Land Cover or Dynamic World by focusing on the change of certain phenomena or type. This focus adds a different dimension of information, as it shows how the Earth's surface has changed over time.

Land Cover Flows is a dataset published by the European Environment Agency (EEA) in 2020. It follows the main drivers of change across 38 EEA member states and the United Kingdom. The currently downloadable dataset observed changes between the years 2000 and 2018. It is derived from Corine Land Cover and interprets 1892 possible one-to-one changes between 44 different Corine Land Cover classes. The dataset observes changes like urban expansion, deforestation, and reforestation through 9 main classes of change and more subclasses. It is the most comprehensive dataset of change ever created in Europe (URL 3).

The United States Geological Survey (USGS) counterpart to the European effort is Land Change Monitoring, Assessment, and Projection (LCMAP). This project provides several types of change data in numerous product categories, separated into grids over all contiguous United States between the years 1985 to 2021. Datasets are downloadable from USGS and include several CD products, such as:

- Annual Land Cover Change: a synthesis product derived from the primary land cover.
- Time of Spectral Change: identifies the temporal origin of the changes defined as "breaks" where spectral observations have diverged from model prediction.
- Change Magnitude: provides information on the spectral strength or intensity of the time series model "break."
- Time Since Last Change: represents the time, in days, from either the product publication or the last "break." The time series only considers the first change of the

year for the whole year, so if there are more changes within a year, a new change will be recorded in the next year.

The project uses Landsat collection data and U.S. Analysis Ready Data for classification using a boosted decision tree classifier and time series modeling to monitor changes. The problem with LCMAP is that it is path-dependent with the Landsat 1 data, making innovation harder. Using multiple collections is discouraged, and using new Landsat observations may result in changes to the model. (USGS 2022)

Land use is not the only type of CD dataset. Other notable focused datasets focus on the state of vegetation. Notably, the Global Mangrove watch observes the net change in the mangrove area to help conservation (URL 4). Another vegetation focused change dataset is the Global Forest Change made by the Global Land Analysis and Discovery (GLAD) project. GLAD uses Landsat data and time series to produce a series of binary maps depicting changes in forest cover between the years 2000 and 2023. The maps are able to show the results of disasters' impacts on forests, just like in the case of the 2005 extratropical cyclone Gudrun that decimated the southern Swedish temperate forest. Deforestation in the Brazilian Amazon is also a highlight of this change dataset, as it is one of the most aggressively deforested places on Earth (Hansen et al. 2013).

Global Forest Watch (GFW) is a global institution that operates in numerous countries. It functions as an initiative dedicated to near real-time monitoring of the world's forests to promote conservation and mitigate deforestation. GFW provides openly accessible maps with data visualizations for public use with the aim of supporting conservation and mitigating deforestation.. Users can further contribute by reviewing observed forest cover changes and uploading ground-truth data, such as photographs or areas, through a mobile application called the "Forest Watcher". These changes can be interpreted through online web maps powered by Planet and Google Earth Engine or accessed on the Data Hub, which offers all data as open access.

Currently, GFW incorporates over 170 diverse datasets from various sources and scales. These dataset sources range from local forestry ministries to other relevant institutions. However, near-real-time monitoring is facilitated by dedicated alert services, such as the

RADD (Radar for Detecting Deforestation) a sentinel-1 based system and GLAD alert systems, which will be further elaborated upon later in this chapter. Additionally, previously mentioned Global Forest Change is also incorporated in the web map (URL 5).

Apart from land use change, there are not many change datasets focused on anthropogenic phenomena. An exception is the Atlas of Urban Expansion, which performed post-classification analysis using Landsat 5 through 8 on various cities across the globe. It tracked built-up area and road changes between 1990 and 2014 by sets of "from-to" maps for each city. The data is publicly available, but unfortunately, the project data ends in 2013. The project was able to detect the significant changes occurring in expanding cities in developing countries. (URL 6)

Most of the previous examples are based on supra-yearly or even temporally isolated datasets. Terra-I is not one of them. This project tracks vegetation changes resulting from human activities in near real-time, providing updates every 16 days across the Latin America. The system operates on the premise that natural vegetation follows a predictable pattern of changes, which can be monitored using MODIS data. A neural network is trained to understand the normal pattern of changes in vegetation greenness in relation to factors such as rainfall and elevation. NDVI is used to calculate a baseline, and deviations from this baseline are identified as changes. Areas that do not follow the expected pattern, as predicted by the model considering rainfall and seasonality, are labeled as undergoing change. (Reymondin et al. 2012)

Another notable sub-year CD product developed by the Global Land Analysis and Discovery (GLAD) project is the DIST-ALERT: Near-Real Time Disturbance Alert. This product provides near real-time alerts of potential disturbances, measured as percentage loss in vegetation cover. Similar to Terra-I, the DIST-ALERT algorithm establishes a baseline for normal vegetation activity using data from the past three years within a 31-day window. Disturbances are identified by a secondary algorithm that evaluates spectral distance to filter out non-forest changes, such as variations in sun angle. This process utilizes Harmonized Sentinel-2 data and Landsat data. With a multiday resolution, the DIST-ALERT system has been operational since 2022 and forms part of the Observational Products for End-Users from Remote Sensing Analysis project. (Hansen, Pickens, Song 2024).

3. Change Detection for Detecting Changes Related to Armed Conflicts

Numerous case studies and reviews have examined various propositions and methodologies to implement the most appropriate methods for monitoring war-related damage in urban areas. Monitoring in this case is critical due to the devastating impact of war on civilian infrastructure, cultural heritage, and public safety. Accurate and timely damage assessment is essential for humanitarian aid delivery, post-conflict reconstruction planning, and investigations of potential war crimes. According to the United Nations (UN) (2023), approximately 114 million people are displaced by ongoing wars. An additional 33,000 civilians are estimated to have been killed in conflicts in 2023 alone. Cities, being the centers of society, often become centers of conflict. For example, the Civil War in Syria resulted in 33,500 damaged structures in Aleppo alone, which together with other damages to infrastructure in the city between 2016–2018 would cost 8 billion dollars to repair according to the World Bank's Syria Damage Assessment of Selected Cities: Aleppo, Hama, Idlip (2017).

Currently, there are only a few public monitoring missions or organizations utilizing RS for monitoring war damage. One of them being the UNOSAT which is part of the United Nations Institute for Training and and Research (UNITAR). UNOSAT offers Rapid Mapping Service which allows for governments and non-governmental organizations to access maps and analysis outputs regarding natural and humanitarian disasters. Recent outputs regarding the Gaza conflict are created through manual visual analysis using very high spatial resolution data like Maxar’s WorldView-3 with 30 cm (panchromatic band) to asses building damage, additionally NDVI differencing has also been used for detecting changes in vegetation using the WorldView-2 data with 50 cm spatial resolution (panchromatic band). (URL 7)

Andrew Marx (2016) introduced a method for monitoring urban devastation in Aleppo and Damascus during the civil war. The study uses a methodology predicated on band normalization. The aim was to determine which changes in reflectance would explain the devastation of the built-up area, using Landsat 8 and 9. The central hypothesis is to normalize the images using invariant features, which will then allow the analysis to compare these images over time and find the disturbances that have taken place. "This approach creates an expected value for every pixel of these two cities at every date of the year, based on that pixel's historical baseline. The signal for newly destroyed buildings is detected when their post-destruction pixel value is significantly higher than their expected value for that date" (Marx 2016). This method has its shortcomings, one of them being the noise of the baseline images, which is unaccounted for in the transformed bands, which makes it only marginally more accurate than the other bands. The overall accuracy of the composite band was 74 % when evaluated against the ground truth pixels.

Fakhri and Gkanatsios (2021) used Support Vector Machine post-classification and optical and radar data combination for detection in Mosul City (Syria) and achieved an overall accuracy of 94%. Researchers in this study used Sentinel-1 and -2 data, combining both to accurately classify urban areas and then compare pre- and post-damage in the post-classification analysis (Fakhri, Gkanatsios 2021). Despite the satisfactory results, the study can be faulted in the use-case of only one observed area, which makes it only proven to work at specific imagery and region.

Mueller et al. (2021) focused on Convolutional Neural Network (CNN) techniques trained using human-labeled destruction instances from Syrian cities. Google Earth / Maxar satellite imagery, gridded into a 64 by 64 pixel matrix, served as the dataset. Eventually, random-forest algorithms were implemented as the second stage of the model to increase its accuracy. This model proved to have a 90% Overall accuracy (Mueller et al. 2021). This result is only relative; the model may perform worse in reality. One of the limitations of this paper was the use of Google Earth imagery, which is compiled from many different datasets that could influence the results.

Overall, each presented study has its advantages and disadvantages, but monitoring needs a stable and robust method that will be universal. There are many different aspects of variables that may affect the spectral characteristics of different locations. Therefore, the CD between classifications would be the most effective method to be used universally. Additionally, as machine learning techniques evolve at a rapid pace, convolutional neural networks (CNNs) stand out as holding the most potential, albeit demanding advanced computing and graphics power for optimal performance.

Aside from the UNOSAT monitoring efforts, the conflicts in Ukraine and Palestine have inspired other monitoring initiatives. Notably, O. Ballinger (2023) implemented the Pixel-Wise T-test to detect changes using Sentinel-1 imagery. The analysis focused on the Gaza Strip and its urban areas to create probability maps indicating damage likelihood. The algorithm has been tested and trialed on the 2020 Beirut explosion. However, in the case of Gaza Strip, the algorithm achieved around 80% overall accuracy. The results, however, did not come without their caveats. Due to the nature of the algorithm, older changes were more likely to be detected, and non-war-related changes, such as dynamic tent cities, were detected as damage. Results are available through a Google Earth Engine web app for the public to view, along with geolocated photographs and videos pinpointed on the map. (URL 7)

A similar approach is utilized by C. Sher and J. V. D. Hoyek. In their assessments, SAR data was also utilized in Sentinel-1 imagery. Their approach is based on the interferometric synthetic aperture radar workflow. Furthermore, their method relies on tracking changes in coherence, an indicator of structural stability and presence, across many years of data. Likely damaged areas are then post-processed to remove false positives. This approach also excludes non-urban areas and is cross-referenced to individual buildings, providing sub-weekly reports. Although these assessments had tremendous success in media reports, the official methodology or approach has not yet been published, and hence accuracy has not yet been determined. (ESCWA 2024)

4. IR-MAD

In relation to the previous chapters, IR-MAD can be considered a transformation-based CD method that produces binary maps, which are pixel-oriented, calculated from multispectral or hyperspectral data. With that categorization IR-MAD is on the robust side of methods, reducing redundancy and obtaining new information derived from spectral information of the bands.

Before the creation of IR-MAD, only multivariate alteration detection (MAD) was available, A. A. Nielsen developed it in 1997 to showcase a new robust method to compare the traditional methods at the time such as PCA. MAD transformation answered the issue of differencing images for CD, as simply differencing images can result in changes due to varying radiometric conditions rather than actual changes. By being invariant to linear scaling, MAD is insensitive to differences in gain settings in measuring devices.(Nielsen, Conradsen, Simpson 1998). MAD uses linear transformation and canonical correlation to incorporate change information by subtracting the canonical variates derived from the linear transformation administered through the Canonical correlation analysis (CCA) (Nielsen, Conradsen, Simpson 1998). However, MAD itself was due to its straightforwardness susceptible to outliers which may show false changes due to atmospheric differences or noise. The incorporation of iterative reweighting (IR-MAD) aims to consider outliers and thus mitigate the impact of noise and other spurious effects by improving the no-change background by giving higher weights to the observations of no-change in the Canonical Correlation statistics. (Nielsen 2005)

IR-MAD has been used in numerous studies as a supplementary method to synthesize with other methods in order to obtain the desired results. For example, IR-MAD was employed to depict information about urban sprawl in Wuhan City, China, and was subsequently integrated with other methods to enhance the accuracy of urban CD. IR-MAD was specifically chosen because of its ability to handle noise and diverse urban spectral characteristics and compared to other methods such is CVA, whilst detecting changes with the use of pair of GaoFen-2 VHR multispectral imagery (Luo, Liu, Wu, Guo 2018). In another study regarding urban sprawl in Germany, IR-MAD was used to supplement the classification of changes in land use in a time series by finding no-change areas between Surface Reflance Landsat Imagery (Ghazaryan et al. 2021).

Furthermore, IR-MAD has been tested on non-urban settings as well, notably the deforestation. La Barreda-Bautista et al. (2011) compared IR-MAD with post-classification techniques to detect changes in land use, deforestation and regeneration of forests on Landsat imagery. IR-MAD showed promising results in detecting changes and mitigating false changes as opposed to conventional post-classification methods. In a similar study IR-MAD has been utilized together with post-classification analysis to find binary changes in an Indonesian rainforest in order to find more subtle classes for classification which would not be normally detected by general land cover classes. (Panuju, Paull, Trisasongko 2019)

4.1. MAD and Canonical Correlation Analysis procedure

The Multivariate Alteration Detection (MAD) procedure, developed in 1997 by Nielsen and Conradsen, utilizes Canonical Correlation Analysis (CCA), a classical statistical analysis, to enhance the detection of changes across different images. The essence of using CCA is to maximize the similarity between the linear combinations U and V of the two original image sets X and Y. This process ensures that genuine changes become more apparent in the difference image. The MAD components themselves are the scalar differences of the transformed image bands of U and V, encapsulating the change information in a single image. Due to the transformation of both images MAD does not strictly demand pre-processing of the images as the procedure alone can be used for normalization.

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{pmatrix}, \quad Y = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{pmatrix}$$

(1)

Before the detection, let's consider two *N*-band optical/infrared images (denoted as X and Y) of the same scene acquired at different times, between which ground reflectance changes have occurred at some locations but not everywhere. Each pixel is multidimensional, having more than one band (1 to *N*), so we can consider each pixel as a random vector (pixel vector) just like in depiction (1). For a CD set of bandwise differences, it would be possible to create a change vector with low absolute values representing non-change values and high values demonstrating change. However, as Nielson (1998) suggests, simple differencing only makes sense if the imagery is normalized, on the same scale, and calibrated over time. Additionally, subtracting multiple bands at once is difficult to visualize when there are more than three bands. To address this issue and emphasize changes across various spectral bands, a linear transformation that maximizes a measure of change, such as variance, can be applied. Nielson (1998) describes a parameter-rich measure of change that allows for different coefficients to multiply and combine more than the spectral bands in the way that the change information is invariant to linear scaling. This measure is depicted in the linear combinations of U and V in equation 2. Here, all intensities across all N bands in the first image (X) and the second image

(Y) are combined and multiplied by vector coefficients for each image and band so that the correlation between the two combinations U and V, defined as:

$$U = a_1 X_1 + a_2 X_2 + \cdots + a_N X_N = \mathbf{a}^\top \mathbf{X}$$
$$V = b_1 Y_1 + b_2 Y_2 + \cdots + b_N Y_N = \mathbf{b}^\top \mathbf{Y}$$
(2)

The symbol $a^\top$ represents the transpose of the column vector $\mathbf{a}$, transforming it into a row vector $a^\top = a_1, a_2, \ldots, a_N$. Transposition allows the coefficient vectors to multiply with the pixel vectors. Once these linear combinations are formed, the change information is captured in their scalar difference $U - V$ where high absolute values are maximum change areas and values close to zero can be considered no-change areas. Consequently, this change information is consolidated into single images, rather than being dispersed among all N bands. The vector coefficients $\mathbf{a}$ and $\mathbf{b}$ still have to be determined in an appropriate manner. Nielsen (1998) suggests applying the CCA. This method finds a linear combination of U and a separate linear combination V, that maximizes the correlation (3).

$$\rho = \frac{\mathrm{cov}(U, V)}{\sqrt{\mathrm{var}(U)}\sqrt{\mathrm{var}(V)}}$$
(3)

The resulting maximized correlation between U and V is called the Canonical Correlation. The coefficients are computed so the canonical correlation is maximized. However any arbitrary multiple of U and V would have the same correlation since the multiples would cancel out in both the numerator and the denominator of the pairwise correlation coefficient formula (3), therefore, a restraint must be applied and a convenient one is to request unit variance for U and V.

$$\mathrm{var}(U) = \mathrm{var}(V) = 1$$
(4)

With the constraint (4) applied to the CCA, the covariance of U and V can be directly maximized. The coefficients are then found by solving each eigenvalue problem for each pair

$$M_i = U_i - V_i, \quad i = 1, \ldots, N,$$

of bands. Solution of the eigenvalue problems generates new multiband images and the components of which are called the Canonical variates (CVs). The CVs are ordered by their pairwise correlation rather than by wavelength, therefore, the pair of U_1 and V_1 is the most correlated pair and the next pair of U_2 and V_2 is the next most correlated pair, with the added constraint that U_2 and V_2 pair is uncorrelated (orthogonal) to U_1 and V_1. This process continues such that each subsequent pair U_i and V_i is the most correlated pair subject to being uncorrelated with all previous pairs.

Finally MAD variates are introduced as the difference between transformed pairs (eg. x). Where i equals the order of the CVs and the MAD based on the correlation and the N equals the number of pairs/bands. The MAD variates are also ordered by correlation, making the first variate showing maximum similarity and minimum change and the second maximum similarity while being uncorrelated to the first MAD variate. In this way, MAD varieties are essentially uncorrelated difference images where each new image shows maximum difference (change) under the constraint of being uncorrelated with the previous ones. MAD values nearing zero are expected to show minimal or no change, whereas higher values are associated with higher chance of the fact that change has occurred.

4.2. Iteratively Reweighted Multivariate Alteration Detection (IR-MAD)

MAD variates are susceptible to detecting uninteresting changes due to noise or arbitrary spurious differences. The iteratively reweighted extension to MAD was developed by Nielsen (2005) to establish an increasingly accurate background of no change against which to detect actual change, thereby mitigating the effect of uninteresting changes. This is achieved by assigning higher weights to observations of no change in the calculation of the coefficients in canonical correlation analysis (CCA).

$$Z = \sum_{i=1}^{N} \left(\frac{M_i}{\sigma M_i} \right)^2$$

(5)

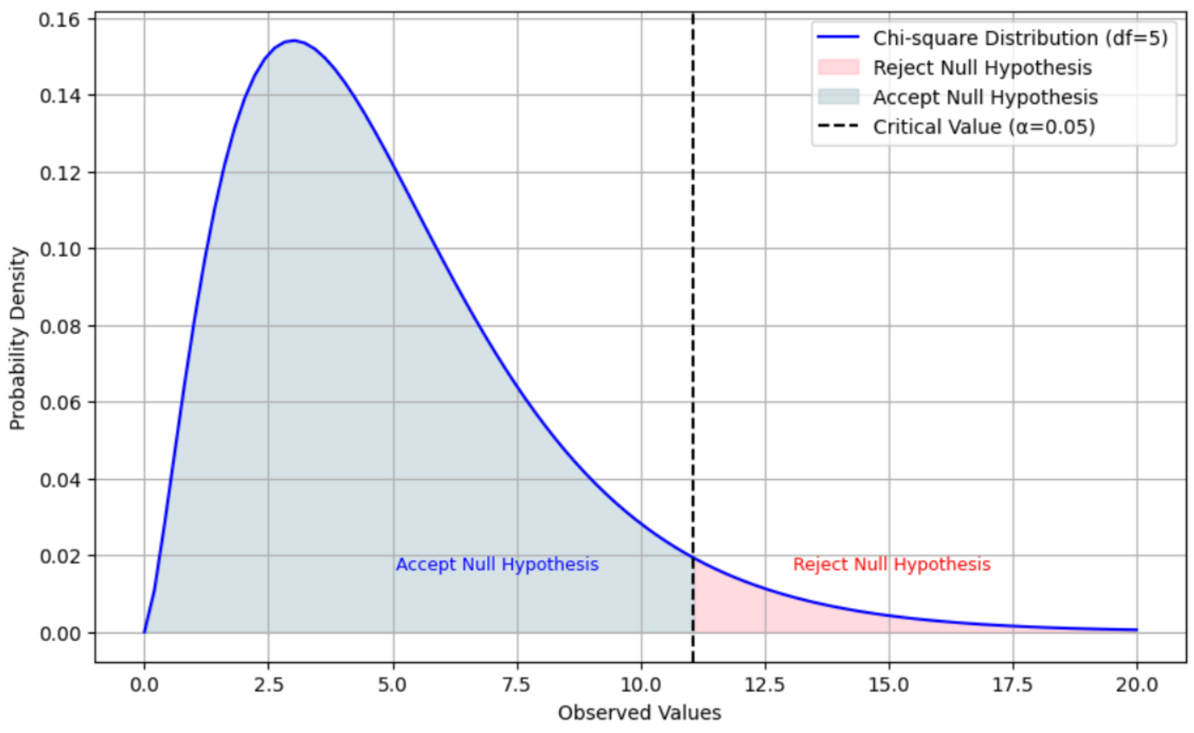
Assuming no changes in ground reflectance, the differences between the CVs would be due to random noise and fluctuations, which are normally distributed. As a result, the MAD variates, being (orthogonal) uncorrelated, should follow a zero-mean normal distribution due to the Central Limit Theorem. Hence, under that assumption, the sum of squared standardized MAD variates represented by *Z* in equation (5) should approximately follow χ^2 distribution with *N* degrees of freedom. Therefore, with the standardization, *Z* can be interpreted as the likelihood ratio test statistic for change where null hypothesis equals “no change” and alternative hypothesis “change”. Likelihood ratio test statistic is then used to derive the chi-square distributed test statistics. Consequently, *p* values for an observation *z* from the χ^2 distribution with *N* degrees of freedom are calculated as shown in equation (6). Where *z* is the observation; the $P2; N(z)$ is the cumulative distribution function and (1 -) defines the right tail test which is synonymous with likelihood ratio test.

$$p(z) = 1 - P2; N(z)$$

(6)

With calculated *p* values can label observations based on their placement in the distribution. In the context of IR-MAD the *p* value represents the probability that the observation (*z)* would occur if the null hypothesis of no change would be true . Thus small *p* values are associated with change as they bear lower probability of following the χ^2 distribution. IR-MAD uses the *p* value itself to weigh each pixel before re-sampling to determine the means and covariances for the next iteration. Larger weights are given to the pixels which have higher *p* values and smaller weights to pixels with small *p* values, thus gradually reducing the influence of the change observation in the MAD transformation, and therefore establishing a better “no-change” background. The process repeats until an optimal no-change background is established or based on the user-chosen number of iterations. With each iteration there will be higher canonical correlations and therefore higher maximization of variance of the difference images. (Canty 2019, p. 390)

Figure 1: Example of χ^2 distribution with 5 degrees of freedom with the Right Tail test with significance Level of 0.05



Source: Author's work

Once the "no-change" background is achieved, the final sum of squared standardized MAD variates (Z) can be used for thresholding to create a binary image using a right-tail test. In Figure (x), we see an example of the right-tail test of a χ^2 distribution with 5 degrees of freedom. The significance level (alpha) is set to find a critical value at which the null hypothesis of "no change" is rejected in favor of the alternative hypothesis of "change." The alpha is set at 0.05, meaning there is a 5% risk that a pixel will be classified as a change even though it is actually no change. *P* values are compared to the alpha, and if they are equal to or smaller than the alpha, the null hypothesis is rejected. Binary layers derived from the critical value can be created to show the "changes".

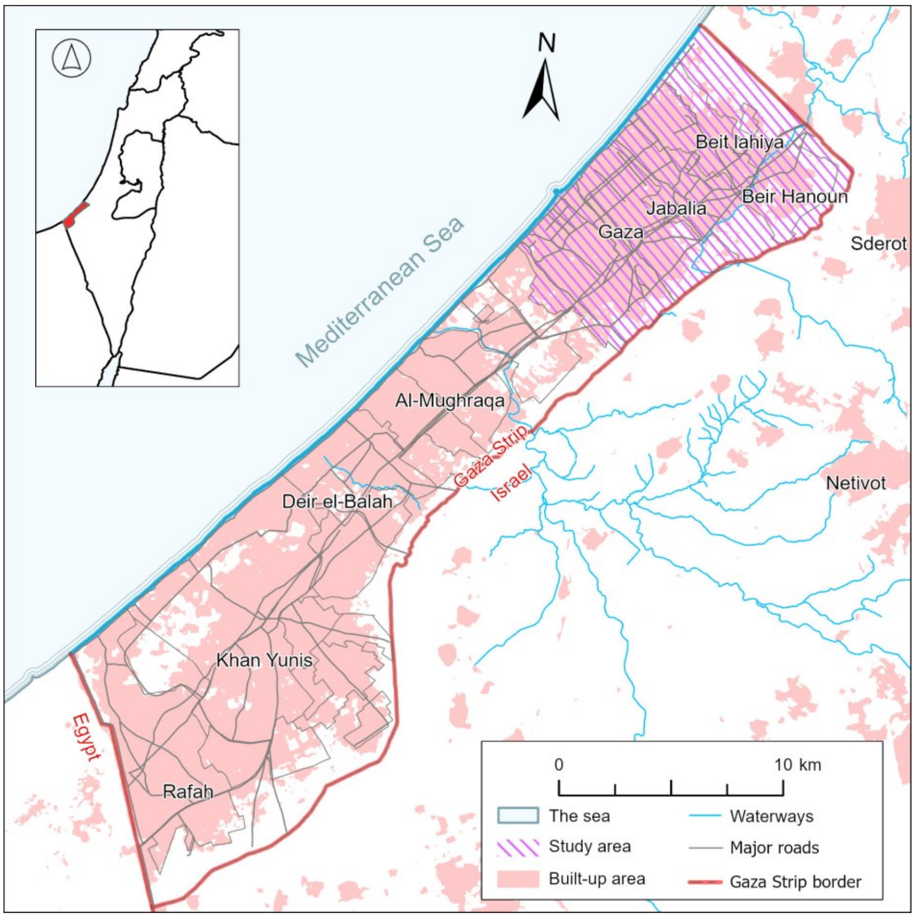
5. Study Area, Data and Software

This chapter outlines the tools, software, and study area that enabled the CD analysis. Firstly, a study area was carefully selected to test the method. This chapter then details the characteristics of the Sentinel-2 and Planetscope satellites, which provided the data for the analysis. Lastly, it highlights the pivotal tools used in the analysis: the Google Earth Engine (GEE) Application Programing Interface (API) and the Visual Studio Code. These tools were integral in processing the satellite data and performing the necessary computations for the CD analysis.

5.1. Study Area: Gaza City urban area

Gaza City and the entire Gaza Strip are among the most recent areas affected by warfare. The Gaza Strip has an elongated shape and is densely built up, as shown in Map-1. Unlike Israeli settlements, Gaza's land cover is heavily urbanized. For this study, Gaza City and its surrounding urban area were selected for analysis, as Gaza City became the epicenter following the attack by Palestinian armed groups on nearby Israeli towns and civilians. In response, the Israeli military declared a "state of war alert" and moved its troops into Gaza, resulting in widespread destruction. The conflict is estimated to have caused 39 145 fatalities, displaced 1.9 million people internally, and damaged 60% of residential buildings as of July 24, 2024 according to the UN (URL 8). The total area of interest is roughly 110 km^2. Its compact size, compared to the entire Gaza Strip, allowed for a more thorough analysis and accuracy assessment representative of the condition of the whole Gaza Strip. The area of interest (AOI) is defined by the administrative boundaries of Beir Hanun, Beit Lahiya, Jabalya, Umm an Naset, and Gaza, accessed through the Humanitarian Data Exchange (URL 9).

Figure 2: Location of Gaza Strip and Study Area



Source: Author's work 2024

Data sources: OpenStreetMap contributors, Natural Earth and Sentinel-2 10-m Land Use/LandCover Time Series Downloader

5.2. Data

In this research, Sentinel-2 multispectral imagery and PlanetScope optical imagery were used for analysis and validation. The final output was derived solely from Sentinel-2 multispectral imagery, utilizing most of its bands capturing a range of wavelengths that facilitate IR-MAD analysis. PlanetScope, with its 3-5 meter resolution, served as a suitable candidate for validating the IR-MAD CD due to Sentinel-2's lower resolution of 10-20 meters.

5.2.1. Sentinel-2

Sentinel satellites, operated by the European Space Agency (ESA) as part of the Copernicus programme, provide continuous Earth observation data. Sentinel-2, having the highest resolution among publicly accessible satellite data, was selected as the primary data source for the CD in part due to its analysis availability.

The Sentinel-2 mission consists of two identical satellites (Sentinel-2A and Sentinel-2B) in polar orbits, following the same sun-synchronous path 180° apart. Before the launch of the second Sentinel-2B satellite in 2017, the sole Sentinel-2A satellite was capable of achieving global coverage in 10 days. However, the addition of a 180° satellite helped reduce this range to approximately 5 days. The mean orbital altitude is 786 km, with an inclination of 98.62°. This sun-synchronous orbit ensures consistent illumination conditions for image acquisition, allowing the satellites to pass over specific locations on Earth at the same local time (URL 2). Both Sentinel satellites carry the MultiSpectral Instrument (MSI), which provides the 13 bands of the Sentinel-2 product in the range of 0.443 to 2.19 micrometers and a focal range of 290 km. The complete range of bands and their central wavelengths can be seen in Table 4.

Table 4: Sentinel-2 mission parameters

Sentinel-2 Bands	Spatial Sample Distance (m)	Central wavelength (nm)
1 - Coastal aerosol	60	443
2 - Blue	10	490
3 - Green	10	560
4 - Red	10	665
5 - Vegetation red edge	20	705
6 - Vegetation red edge	20	740
7 - Vegetation red edge	20	783
8 - NIR	10	842
8a - Vegetation red edge	20	865
9 - Water vapour	60	945
10 - SWIR - Cirrus	60	1380
11 - SWIR	20	1610
12 - SWIR	20	2190

Source: (ESA Standard Document: Sentinel-2 User Handbook 2015)

5.2.2. PlanetScope

The PlanetScope constellation operated by Planet Labs consists of over 130 Dove satellites in sun-synchronous orbits at altitudes between 475 and 525 km. The distributed nature of this constellation allows for near-daily revisits of any location on Earth, with equator crossing times between 9:30 and 11:30 am local solar time. The orbit inclination of 98° ensures consistent illumination for image acquisition.

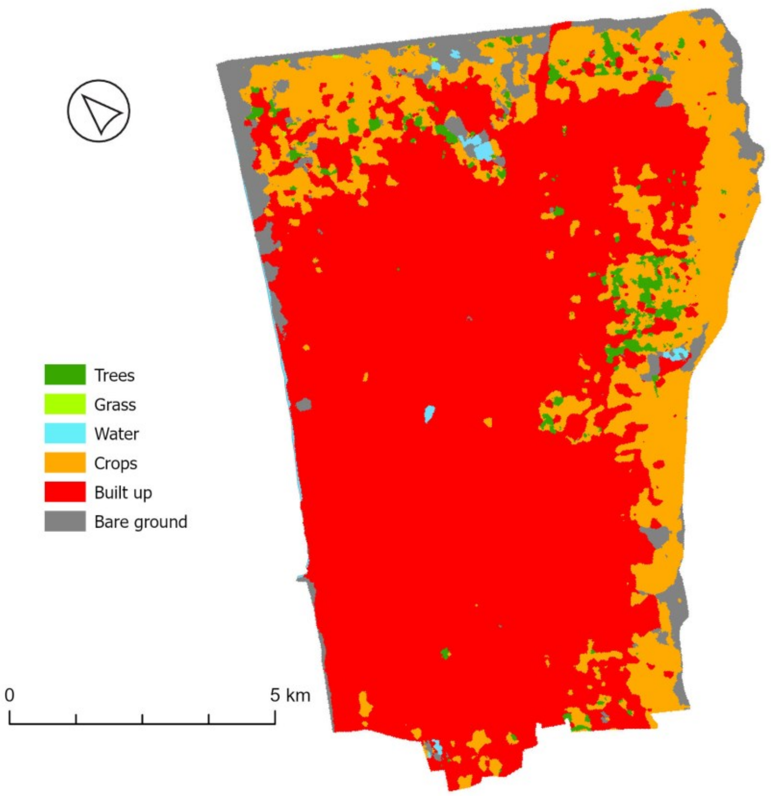
Each Dove satellite carries a multispectral imager, capturing imagery in four standard bands (Blue, Green, Red, and Near-Infrared), with some capturing additional bands (Green I, Red Edge, Yellow, and Coastal Blue). The 3-5 meter spatial resolution, combined with the high temporal resolution (almost daily), makes PlanetScope data ideal for monitoring rapid changes on Earth's surface. In 2021, Planet Labs refreshed the original Dove satellites with a next-generation model, diversifying the PlanetScope product line. Currently, three main categories of PlanetScope data are available: Visual Scene (RGB), Analytic (8 band), and Basic (4 band), each with an orthorectified option.

While PlanetScope's spectral range is narrower than Sentinel-2's 13 bands, its finer spatial resolution and daily revisit frequency provide a unique advantage for applications requiring high temporal and spatial resolution. Additionally, PlanetScope data is accessible through a student program on the Planet Labs website, making it a complementary choice alongside Sentinel-2 in this research (URL 10).

5.2.3. Dynamic world

The Dynamic World land cover product, developed by Google in collaboration with the World Resources Institute, provides near real-time, high-resolution land cover classification. Utilizing deep learning models with a highly scalable cloud-based system, it continuously classifies land cover into nine distinct categories: water, trees, grass, flooded vegetation, crops, shrubs, and scrubs, built, bare, and snow and ice, based on Sentinel-2 satellite imagery. This product offers detailed land cover in near-real time, making it possible to mask Sentinel-2 imagery corresponding with the same capture date. The Dynamic World data follows the temporal resolution of Sentinel-2 imagery, resulting in classification updates every 2-5 days Brown et al. (2022). Figure 3 displays the Dynamic World data for the date corresponding to the pre-war change baseline. This data is derived from the same Sentinel-2 image used in this research to test the IR-MAD technique. It helps establish the pre-war built-up area mask specific to that date. The built-up mask in Figure 3 is quite dominant. However, it groups together different types of built up, such as less dense suburban houses, smaller buildings, and dense high-rise buildings, into the same class. There is minimal presence of water bodies and natural vegetation in the AOI, and bare ground follows the militarized border and coast.

Figure 3: Land use as classified by the Dynamic world on the 27th of September 2023



Source: Dynamic World, Author's work

5.3. Software

In this research two softwares have been utilized. GEE API together with Visual Studio Code have been utilized to undergo the IR-MAD for CD, and ArcGIS Pro has been used for manually classifying reference data for validation.

5.3.1. Google Earth Engine API

Google Earth Engine (GEE) is a cloud-based platform for geospatial analysis. By using Google's computational infrastructure, it lets users access and process vast amounts of satellite imagery and geospatial datasets quickly and efficiently, eliminating the need to download and store massive datasets locally. Analyses can therefore utilize cloud-based computation, mitigating the need for computational power locally. This enables the creation of web hosted and continuous services such as Global Forest Change as mentioned in the Change datasets.

Sentinel, and MODIS, offering a vast catalog. Users can access different catalogs with products from various times. Data can then be analyzed using the GEE API in the GEE code editor, accessible via JavaScript in the GEE code editor or an independent Python environment. The API functions are equivalent in both environments. JavaScript in the GEE platform is generally more intuitive, being integrated with the website's map viewer and visualization tools, while the GEE API for Python offers more freedom to incorporate different Python libraries. The GEE API was used with the web-hosted platform to access tasks and collect imagery prompted from the environment. Outputs were saved into personal projects as assets and later accessed via the export path defined at the beginning of the code. Besides cloud computing of the outputs, a vast array of functions was used in Canty's code for implementing the IR-MAD. Additionally, the K Means Clustering function from the GEE library was utilized to compare with traditional thresholding techniques.

5.3.2. Visual studio code and Python libraries

Visual Studio Code, an integrated development environment (IDE), was utilized to process the API and other libraries. Although Google Colab is cloud-based and more intertwined with Google infrastructure, Visual Studio Code allows access to locally stored data. This facilitated the use of other libraries for visualizing, exporting and calculating statistics for imagery such as scipy or rasterio and easier workflow with Arcgis Pro

Another notable library used for working with the GEE API in Python is geemap, a package for interactive geospatial analysis and visualization with GEE. In working with IR-MAD, geemap was primarily used for visualization, providing capabilities similar to the web-hosted GEE platform, such as image stretching and band RGB compositing. Additionally, the geemap's exporting function was heavily utilized, enabling the local export of GEE imagery at a chosen scale to a local machine.

6. Methodology

Methodology of this research is derived directly from the research objectives, which aims to fulfill with the completing of 4 crucial steps:

1) **Preliminary processes**
2) **IR-MAD execution and thresholding**
3) **Accuracy assessment**
4) **Web map development**

Together, these steps make up the methodology of the CD analysis. The ultimate goal is to derive the most accurate CD of changes occurring in Gaza City between the selected dates, that can be later incorporated into the web application and for the whole Gaza Strip.

6.1. Preliminary processes

6.1.1. IR-MAD implementation and environment selection

The IR-MAD algorithm has been developed for various environments, including Matlab, ENVI, Google Earth Engine (GEE), Python, and even in the ESA Charter Mapper. The most recent implementation combines Python and Google Earth Engine by utilizing the GEE API with Python in an IDE of choice. This approach allows the computation of the algorithm to be done remotely while keeping the code stored locally. The GEE API can be accessed by the Python IDE through the \`import ee\` command and facilitated with \`ee.Authenticate()\` and \`ee.Initialize(project='ee-project_name')\`. Once authorization is complete, users can utilize the GEE API with the same functions available in the GEE website's code editor, although without the embedded visualization tools.

This implementation, along with other remote sensing methods utilizing Python and the GEE API, is utilized in the "Image Analysis, Classification, and Change Detection in Remote Sensing, with Algorithms for Python" by Canty (2019). Tutorials based on the book’s CD chapter were later uploaded to the GEE community (URL 11) and are available as open access with downloadable Jupyter Notebooks. For this research, the "Change Detection in Google Earth Engine - The MAD Transformation (Part 2)"(URL 12) tutorial and its Jupyter notebook was downloaded through available Google Colab and functions programed and described by Canty were processed into separate module so the function run_imad and other instrumental functions can be simply called into the Jupyter Notebook. The functions that form the inner workings of the IR-MAD analysis, as described in Chapter 4, are included in Canty's tutorial and served as the foundation upon which the rest of the code was reprogrammed to suit the specific needs of this research. That included, apply_mask, export_image and collect. These functions were relatively simple and provided means of manipulating the data.

6.1.2. Obtaining imagery

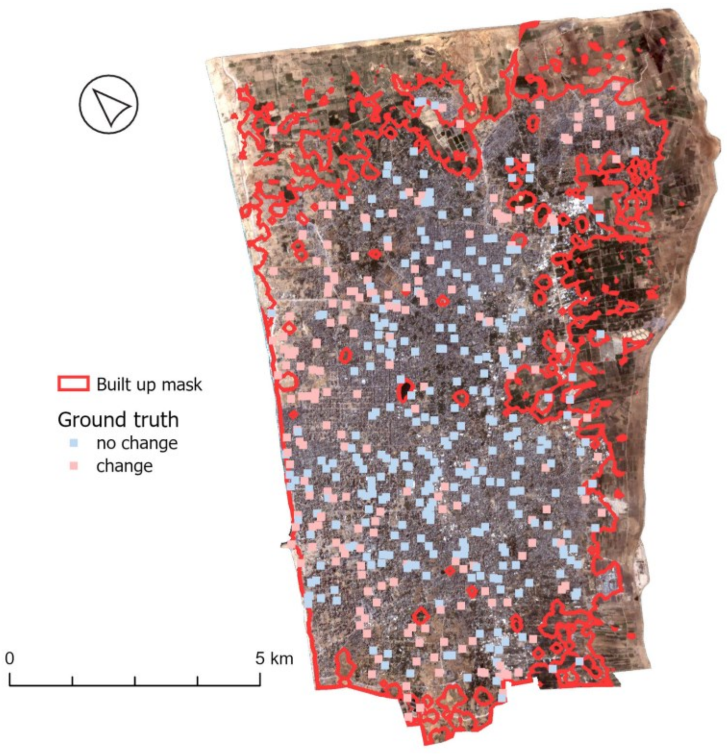
For the IR-MAD two multi-band images of the same scene acquired at different times, between which ground reflectance changes have occurred at some locations but not everywhere are required. What is not required, is the Surface Reflectance (SR) correction, as IR-MAD does not require so due to its transformation properties. However, for purposes of the research SR will be used to further mitigate the effects of the atmosphere on the imagery. Therefore, Sentinel-2 Level 2A images were used for the analysis. Location in question was Gaza city and its surrounding. Due to the compact size of the AOI and urban density, dates with no cloud cover were selected for the IR-MAD analysis instead of computing cloud masking, finding these dates was done by visual check and with the use of SCl, QA and metadata of the images. Changes resulted in clouds and their shadow would significantly affect the results, overshadowing the changes on the ground, creating false changes. Firstly, date for detecting changes was collected to compare to the 27th of September as the last date of Sentinel-2s imagery before the conflict that had minimal cloud cover to temper with the analysis, essentially, it's the closest date to 7th of October when the Israeli-Hamas crisis began. This date was collected to be 26th of November, being a cloudless image which coincided with Planetscope data which were later on interpreted for creation of assessment points to test the accuracy.

6.1.3. Pre-processing

Similar to the O. Bellinger's approach (URL 7) of monitoring changes in the Gaza Strip, a choice was made to concentrate exclusively on alterations within the urban area, excluding the surrounding shrubland and water bodies. Accordingly, the Dynamic World land cover product was accessed via the Google Earth Engine (GEE) API to mask out all non-urban areas. Land cover data generated on September 27th (2023) was selected to delineate the built-up area within the study region.

The apply_mask function was programmed to mask Sentinel-2 images prior to executing the run_imad function. The apply_mask function takes a Sentinel-2 image and a binary mask image as inputs, producing a Sentinel-2 image constrained to the masked extent, masking everything but what class is used to make the mask. Mask was extracted from the Dynamic World database via the eq. function which selected the most likely class name of 'built' for the mask. The run_imad function inputs a Sentinel-2 image and binary mask image, resulting in an image with the extent of the mask as depicted in Figure 4.

Figure 4: Assessment points and the built mask



Source: Dynamic world, Sentinel-2 and Author's work

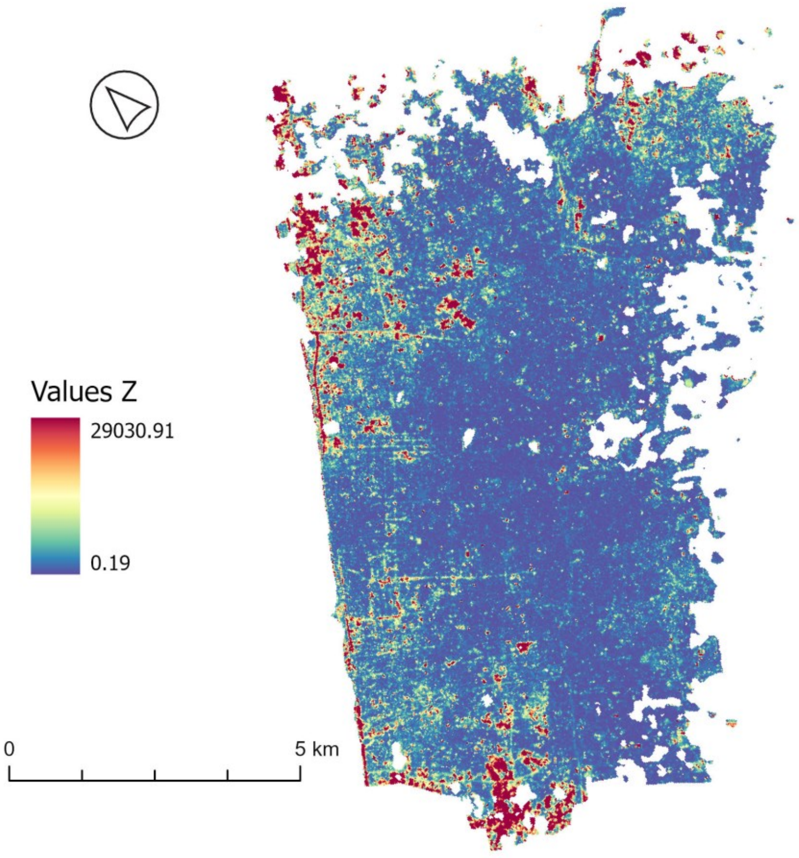
6.2. IR-MAD execution and thresholding

To determine the most accurate binary change maps, different band combinations and thresholds were compared. Initially, downsampled 10-m band combinations were evaluated together, followed by 20-m layers. Within the 20-m and 10-m layers, visible and infrared bands – B2, B3, B4, B8, B11, B12 were utilized for the analysis and comparison.The spectral bands with the highest resolution as well as the SWIR bands were selected. The coastal aerosol band (B1), which is used to monitor chlorophyll concentrations in water as well as phytoplankton and algal blooms or for atmospheric corrections and unsuitable for urban non-vegetation CD analyses. As well as the Red Edge bands (B5, B6, B7 and B8A), which are more suitable for looking for changes in vegetation, forest ecosystems or agriculture. The choice allowed a trade-off between spectral information and the computational cost that would be added with additional bands. The objective was to identify the most accurate binary layer that encompassed changes between September 27 and November 26, 2023 and whether 10-m sampling for calculating covariances in the IR-MAD is more accurate then 20-m.

Canty's code (2024) was utilized to achieve this, notably, the run_irmad function which inputs the selected images for analysis together with a list containing bands to undergo the analysis. Canty's code allows for scaling within the function for covariance and MAD, this scale sets at what scales are the covariance matrices, the scale is later passed into the run_irmad function for execution of the main function. The tested bands were visible infrared bands, but any combination of bands could be used. Once the function was initialized, the GEE platform received a task via the export path to the GEE project, which was specified as a variable in the code. This task can be monitored in the task manager, where it generates an N-band difference layer and an image of squared standardized MAD variates, this image is appended as the last band following the MAD variates. The resulting layer was then accessed via the export path in the code, and the new MAD variates were selected along with a new image (im_z) that selected the *Z*-image using the .select tool. The *Z* is the sum of square standardized MAD variates and is expected to follow the chi-square distribution. Once im_z was obtained, the degrees of freedom were calculated by subtracting one from the total number of bands. With the degrees of freedom known, level of significance was used to test for change.

One of the disadvantages of many transformation and algebraic methods such as IR-MAD, PCA or CVA is often the need to distinguish changes from no changes manually or with other means such as thresholding, which is a costly task that often relies on vast amounts of time spent trying out different optimal thresholds(Lu et al. 2004). However, various algorithms have been developed for thresholding, but the amount of methods is vastly different for various tasks. Therefore, simple effective and thresholding methods often come from universal techniques such as the unsupervised change classification suggested by Canty (2019). In his example the unsupervised algorithm used gaussian cluster changes for aggregating the information to various classes. However, in the MAD tutorial (2024) K-means tutorial was used. Due to the availability from the tutorial K-means clustering was used to find the binary layers. Additionally, IR-MAD allows for thresholding based on its statistical properties. This approach is derived from the iterative reweighting procedure of the IR-MAD and from Canty's chapter on IR-MAD in his book where one of the example outputs had the significance level set at 0.0001 for individual variates in order to highlight changes. However, in this research a binary layer is a desired result and therefore the threshold will be executed over the final sum of squared standardized variates (Figure 5), and in order to be rigorous the level of significance was set at 0.00005 to assure a robust binary layer.

Figure 5: Sum of squared standardized MAD variates stretched at 2 deviations



Source: Author's work

In the context of this approach, errors can occur in two main forms: Type I and Type II errors. Type I Error (False Positive) occurs when the null hypothesis (no change) is incorrectly rejected, leading to the labeling of pixels as showing change where there is none. Conversely, Type II Error (False Negative) happens when the null hypothesis (no change) is incorrectly accepted, resulting in the failure to label pixels as changed where a change actually exists. The chosen significance level of 0.00005 means that any *p*-value in the image less or equal 0.00005 is considered statistically significant, indicating a change. The *p*-value image represents the probability of observing the given data (or something more extreme) under the null hypothesis of no change, and is the same image that has been used to determine the iterative reweighting in IR-MAD. Therefore, a lower *p*-value suggests stronger evidence against the null hypothesis. By setting the significance level at 0.00005, a stringent criterion is set: only changes that are highly unlikely to have occurred by chance (less than 0.005% probability) are distinguished. Essentially such a strict level should mitigate false positives which in context of war-related changes monitoring seemed more appropriate. This is done by using the 'noChangeMask' image that has been demonstrated in Canty's tutorial (2024). Using the selected significance level and degrees of freedom, the critical value was determined using the Omnicalculator's Critical Value Calculator (URL 13) at 27.29 for the Test statistic *Z*, establishing the threshold for distinguishing between no change and change. OmniCalculator, similarly as chi square test sheets, calculates the critical value based on level of significance, type of distribution, type of test (Right-tail) and degrees of freedom. This critical value was then used to create binary layers, which were exported to ArcGIS Pro for evaluation against the ground truth data.

K-means clustering algorithm was utilized to summarize the change information from the MAD variates. K-means clustering is an unsupervised machine learning method that divides data into a fixed number of groups so that the data points within the groups are similar to one another while being different from data points in different groups. The K-means clustering algorithm was accessed through the GEE library under the function ee.Clusterer.wekaKMeans and optimized for generating specified number of clusters while being trained on 50,000 pixels for 20-m data and 200,000 thousand pixels on 10 m data in order to ensure comparatively same training sets (URL 14). First cluster images were generated with 2 classes, and then multiclass clusters were generated to find the ones with best results. The resulting cluster layers were exported into a binary layer and evaluated against the ground truth data.

6.3. Accuracy assessment procedure

Conducting an accuracy assessment for CD is crucial for validating the reliability of detected changes. This involves comparing the detected changes with a reference dataset (ground truth). A common tool for this purpose is the confusion matrix and metrics such as overall accuracy (OA), user's accuracy (UA), producer's accuracy (PA).

OA measures the proportion of correctly identified change and no-change pixels. UA indicates the probability that a pixel classified as change (or no-change) is correct, while PA reflect

To determine the binary map with the highest accuracy and evaluate IR-MADs performance in depicting changes between September 27 and November 26, 2023, an accuracy assessment was conducted. This process required a reference layer with ground truth data. Although UNOSAT provides damage assessments, these are limited to buildings and structural damage, resulting in uneven spatial distribution and neglect of other changes such as craters, debris or soil disruption as the result of heavy equipment. Ideally, sub-meter imagery would be used to create a comprehensive reference dataset, but such imagery is not publicly available. Consequently, a visual analysis of available sources was undertaken.

PlanetScope by Planet Labs provides the highest resolution imagery available to students, with a pixel resolution of 3 meters. Imagery from September 27 and November 26 of 2023 was obtained for accuracy assessment purposes, and was used to visually identify changes between the two dates.. Both images were visualized in ArcGIS Pro, where about 500 assessment points were generated using the Create Accuracy Assessment Points tool and scattered across the urban mask of the study area (Figure 3). And then labeled for change and no change through visual interpretation.

After creating the binary maps, the classified changes were appended to the accuracy assessment points. The Compute Confusion Matrix tool was then utilized to obtain the accuracy metrics. Image of the sum of squared MAD variates was then evaluated based on different *p*-values in order to find the most optimal threshold with highest accuracy. OA, UA and PA were calculated.

6.4. Website Development

Once the optimal approach for detecting changes was deduced, a web map for the whole Gaza Strip was developed in GEE. The aim of the app was to track the changes over time, and therefore, dates, with zero cloud cover covering the Gaza Strip were used across the whole temporal range of the conflict. The dates were processed as in the Accuracy Assessment Procedure and then processed into a series of binary maps, showcasing changes from the first day to dates in between. The binary maps with the highest accuracy were chosen to be displayed in the web map, showcasing the changes that occurred between the dates. The application was published on a website created with Google Sites technology.

The selected images correspond to the following dates: September 27, 2023, November 1, 2023, November 26, 2023, December 26, 2023, January 20, 2024, March 5, 2024, April 4, 2024, May 9, 2024, June 18, 2024, and July 18, 2024. The image selection process and preprocessing mirrored the previously used methodology. However, the final binary change maps were derived from the most accurate method, selected based on a comparison of different pixel and binary layer extraction techniques. The web map was then embedded into a Google Site.

7. Results

The results chapter presents the findings of this research, organized into three subsections. The first subsection details the determination of the threshold through the level of significance, involving statistical analysis to identify critical threshold values that differentiate significant data points. The second subsection outlines the outcomes of obtaining change information through k-means clustering, highlighting data groupings and patterns of change. The final subsection evaluates the types of changes included in the analysis, and the ability of IR-MAD to detect changes related to warfare.

7.1. Binary layers generated by the level of significance

The 20-m dataset results (Table 5) showed a 74% OA in detecting changes against the ground truth data, indicating a balance of false positives and negatives. Out of 485 points evaluated, 221 were correctly classified as 'no change,' matching the ground truth data, while 138 were accurately identified as 'change.' The remaining points were mostly misclassified. The UAs and

PAs were nearly identical, with the lowest accuracy observed in the PA for the 'change' class and the highest in the Producer accuracy for the 'no change' class.

Table 5: Confusion matrix of binary layer produced by IR-MAD on bands B2, B3, B4, B8, B11, B12 sampled to 20-m at 0.00005 *alpha (critical value* ***27.293****)*

	No change	Change	Total	User's accuracy
No change	221	65	308	0.7727
Change	61	138	177	0.6934
Total	282	203	485	
Producer's accuracy	0.7836	0.6798		0.7402

Source: Author's work

In contrast, the 10-m dataset showed lower accuracy than the 20-m dataset when using the same threshold. It achieved 69% OA. The results, compared to those from the 20-m sampling, revealed a noticeable imbalance between Type I and Type II errors, with false negatives being more common than false positives. This discrepancy in error types indicates a tendency to miss actual changes (false negatives) more frequently than falsely identifying changes (false positives), but in both. The overall matches differed by 23 points, indicating worse performance at this scale and higher inaccuracies.

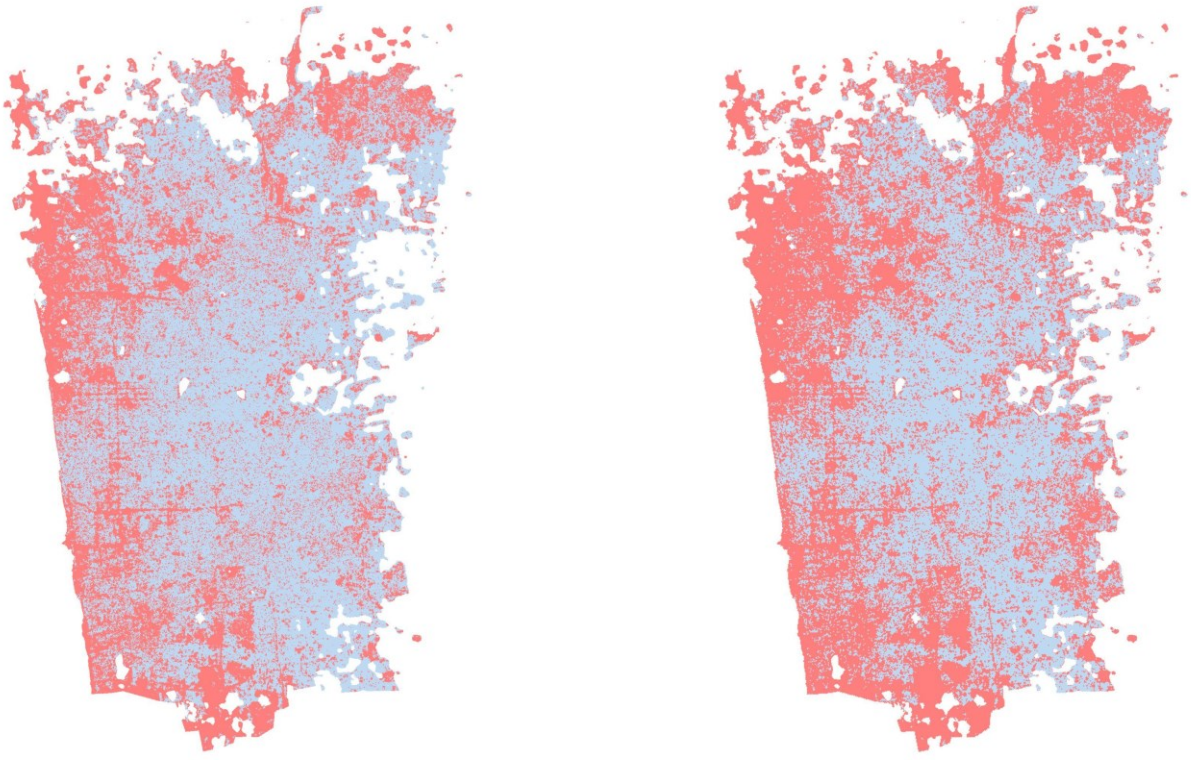
Table 6: Confusion matrix of binary layer produced by IR-MAD on bands B2, B3, B4, B8, B11,B12 sampled to 10-m at 0.00005 *alpha* (critical value 27.293)

	No change	Change	Total	User's accuracy
No change	220	71	308	0.7142
Change	87	115	177	0.6497
Total	282	203	485	
Producer's accuracy	0.7801	0.5665		0.6907

Source: Author's work

Visually, the two binary layers showed significant differences in the area of detected changes as illustrated in binary maps of Figure 6. The binary mask generated from the 20-m sampled data exhibited a dominant change class, whereas the binary mask generated from the 10-m resolution data predominantly indicated the no change class, with detected changes being more isolated. Roads are visible in both of the maps showcasing the changing dynamics of roads.

Figure 6: Binary change maps with 10-m (Left) and 20-m (Right) resolutions.Note: red indicates change, blue indicates no change

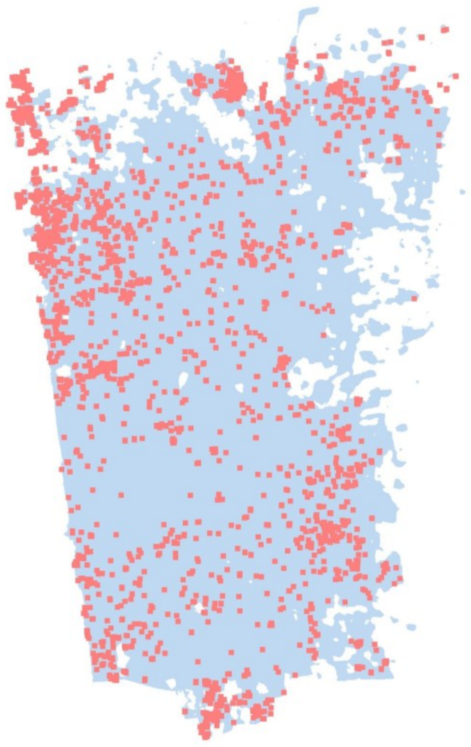


Source: Author's work

Specifically, the IR-MAD analysis using the images with 10-m pixels classified 30 km² as changed, while the 20-m dataset classified 41.09 km² as changed, out of a total mask area of 76 km². These results indicate that either 39.47% (10-m spatial resolution) or 52.75% (20-m spatial resolution) of the area experienced changes between September 27th and November 26th, 2023. For comparison, IR-MAD analysis conducted over a similar time span and the same AOI the previous year (from September 22nd to November 11th, 2022) identified a total change area of 23.5 km² or using the 10-m dataset.

The resulting maps also follow the trend set in the UNOSAT's damage assessment (2023) which examined building damage on 26th of November 2023. In Figure 7, all buildings labeled as destroyed over the built up mask are placed. Showing similarities to the binary masks, especially in the coastal area (left) where IR-MAD detected large swaths of continuous change, and with the relatively unaffected center of the AOI where changes in the binary masks are minimal and destroyed buildings are absent.

Figure 7: UNOSAT damage assessment of buildings; buildings labeled as "destroyed"



Source: UNOSAT's damage assessment (URL 15)

7.2. Binary Layers Generated by k-means clustering

Clustering of the MAD variates into two classes enhanced the dominant class of no-change and minimized the occurrence of false positives at the expense of false negatives. Broader sets of changes using the K-means algorithm were visualized by setting 6 classes, which based on context, were combined as the final binary layer of two classes. Table 9 shows the confusion matrix of the 20-m and 6-cluster image combined into the binary layer. Unlike the 10-m image, the clustering did classified change with more extensive area reaching 71% OA. Higher accuracy results, however, did not indicate the desired simplicity from this approach, and learning for several classes did not find any particular classes of change, except the case 20-m imagery which with 6 classes indicated a stepping stone class between the outlier values and no change values.

Table 7: Confusion matrix of binary layer produced by IR-MAD on bands B2, B3, B4, B8, B11, B12 sampled to 20-m and clustered to 6 classes

	No change	Change	Total	User
No change	279	127	399	0.6817
Change	12	74	86	0.8604
Total	284	201	485	
Producer	0.9557	0.3681		0.7134

Source: Author's work

Image sampled at 10-m was also tested at six classes to capture more change pixels from the two-cluster outputs, as the addition of classes led to noisier no-change clusters as opposed to larger change clusters. With six classes, three were identified as change and three as no change, resulting in a binary layer with 68% OA. The confusion matrix (Table 8) showed a conservative change class with negligible amounts of false positives but a substantial number of false negatives.

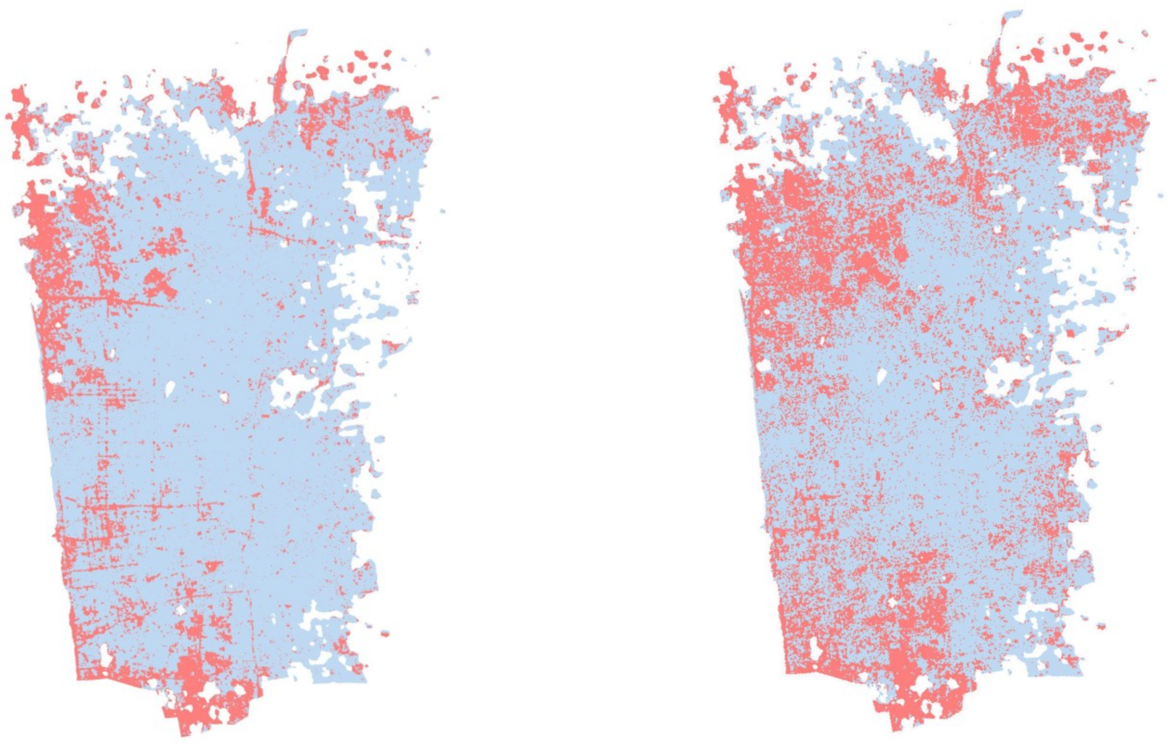
Table 8: Confusion matrix of binary layer produced by IR-MAD on bands B2, B3, B4, B8, B11, B12 sampled to 10-m and clustered to 6 classes

	No change	Change	Total	User
No change	279	150	456	0.6503
Change	5	51	56	0.9107
Total	284	201	485	
Producer	0.9823	0.2537		0.6800

Source: Author's work

The disparity of the multi-cluster derived binary layers at different scales is apparent in their placement of pixels. In Figure 8, the conservative binary layer derived from the 10-m MAD variates and the dominant change layer of the image derived from the 20-m MAD variates can be seen.

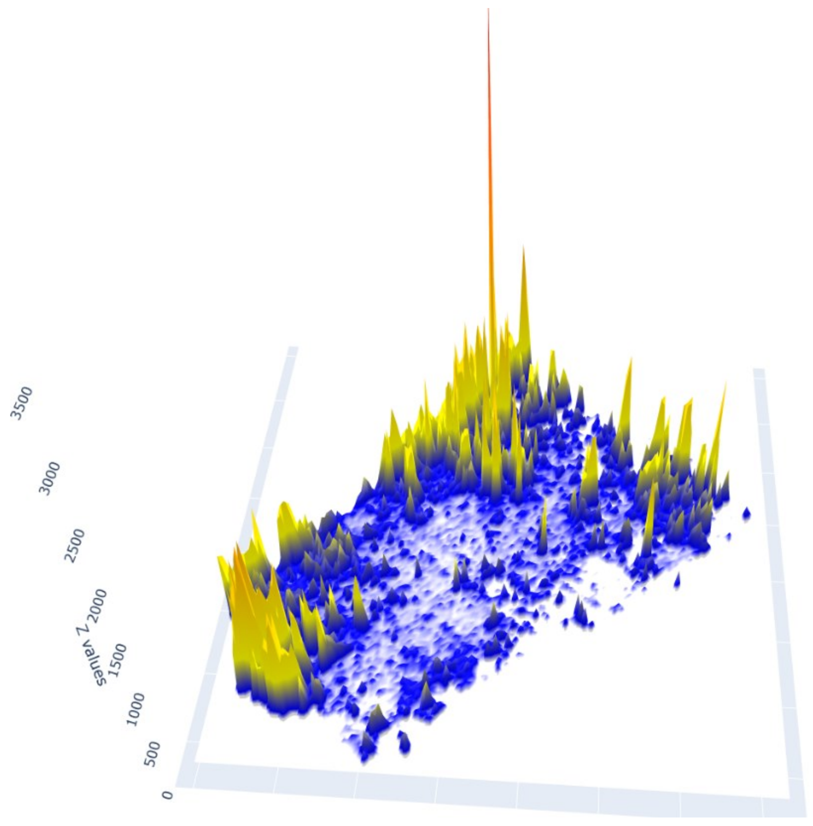
Figure 8: Binary layers generated by multi-class clustering; 10-m spatial resolution from 6 classes (left) and 20-m spatial resolution from 6 classes (right)



Source: Author's work

To further investigate the distribution and peaks of the Sum of Squared Standardized MAD variates (Z) and its effect on clustering algorithms in space, the Z image can be plotted in 3D, and be interpreted for change itself. Figure 9 shows the AOI and its Z values in 3D space. The magnitude of the values Z corresponds with the elevation of the pixel's position within the spatial field.

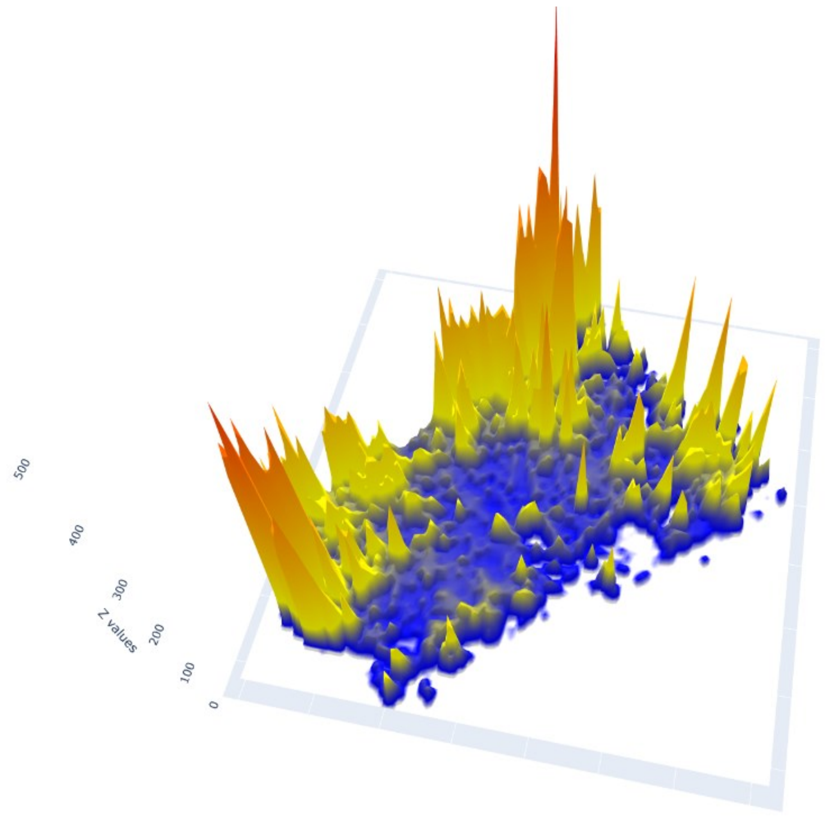
Figure 9: 3D plot of Sum of square Standardized MAD variates generated from 20-m sampling



Source: Author's work

Similarly as in figure 5, the most significant changes are highlighted, but in this case the pixels are shown in aggregates. The original *Z* image was aggregated to suppress outliers, as the highest value of the dataset has over 29,000. Therefore, the units in the image are aggregated 5 per 5 pixels. However, aggregating those did not diminish the magnitude of several changes. The high peaks are notably the surfaces of exposed ground as a result of bulldozing from building outposts and tracks for tanks, and vegetation clearing. It is these places with the yellow peaks that the k-means algorithm clusters to for change with 2 classes. Hence it highlights the fundamental difference between thresholding the image as the k-means cluster around the peaks and similar values to highlight the similar groups of change whilst the threshold will set the same change level across all peaks, however the existence of massive outliers represented by values in the thousands and ten thousands makes the smaller values cling to the no change cluster as the distance is simply way to distant. In comparison, aggregating 10-m pixels into the same size units as the 20-m 3D unit. Resulted in less differentiated values, as more pixels were averaged (Figure 10). Peaks in this case were more similar in their height showing *Z* value peaks that have not been as pronounced as in the 20-m plot.

Figure 10: 3D plot of Sum of square Standardized MAD variates generated from 10-m sampling

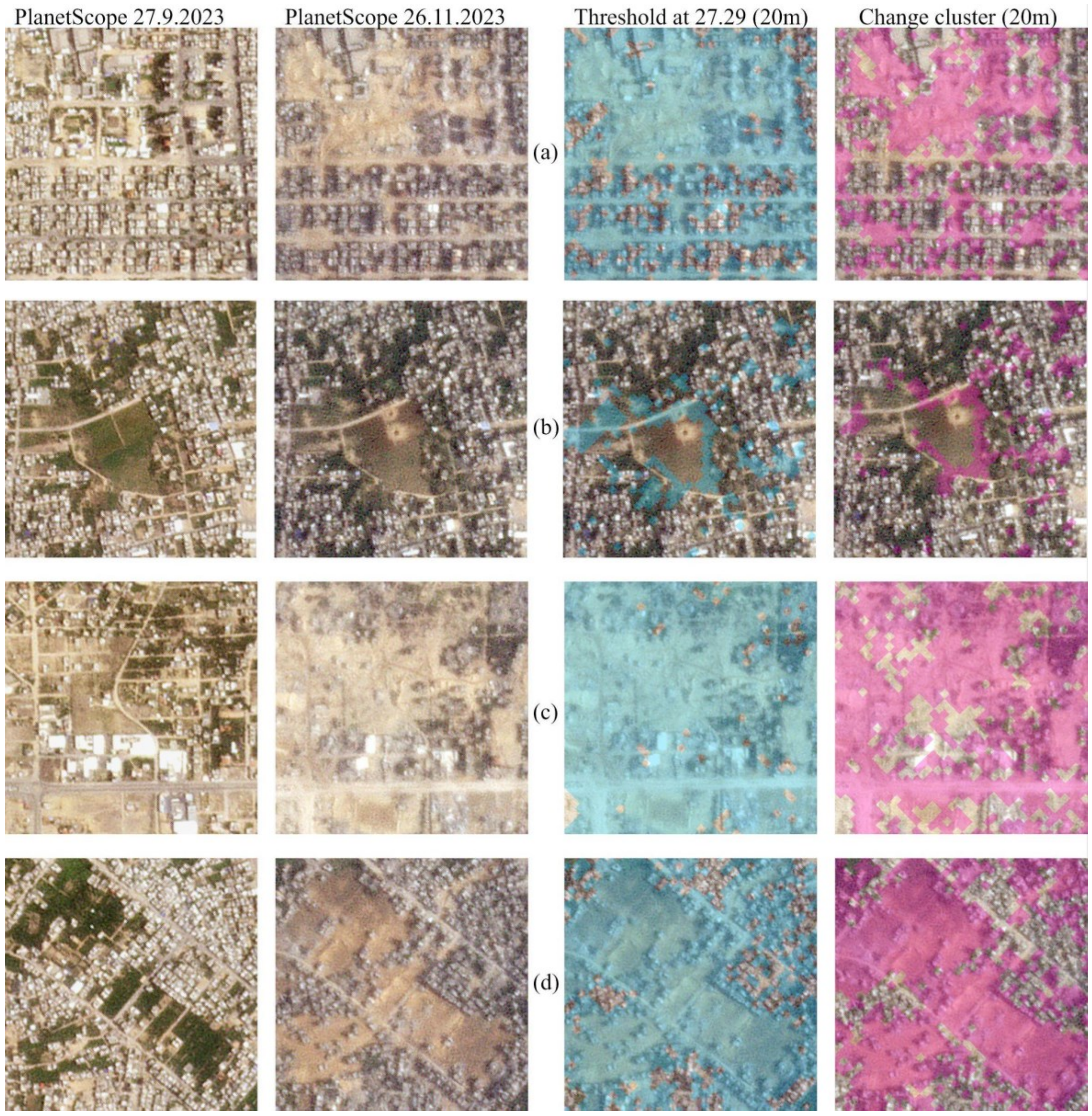


Source: Author's work

7.3. Detection of war related damage

Although not directly distinguished from the main classified change class, noticeable changes related to the effects of war were located. Figure 11 shows examples of different types of damage across the AOI.

Figure 11: Examples of detected changes using the threshold and change clustering.



Coordinates WGS 84 : (a)-[34°27'35"E, 31°32'19"N], (b)-[34°28'43"E, 31°31'11"N, (c)-[34°28'9"E,31°33'20"N], (d)-[34.4857914°E, 31.5422456°N]

Source: PlanetLabs, Author's work

The top row (a) shows inflicted damage and devastation bordering the denser urban area of the Gaza city. In the top right of the image there can be high-rise buildings before the conflict and their destruction following the conflict. In this case both of the approaches for retrieving change information were able to obtain change information, although the clustering of changes has misidentified the affected buildings as no change. Another change observed in the row (a) is the changes of the road network, clearly visible, roads low-lying infrastructure are often covered with debris and dust. It is also the enterpoint into the densely built up part of Gaza city, making it the primary road for tanks and the Israeli army to enter with. The flattened area is also deemed to be the previous Hamas training center and currently an Israeli Defence Force strongpoint, with fortifications.

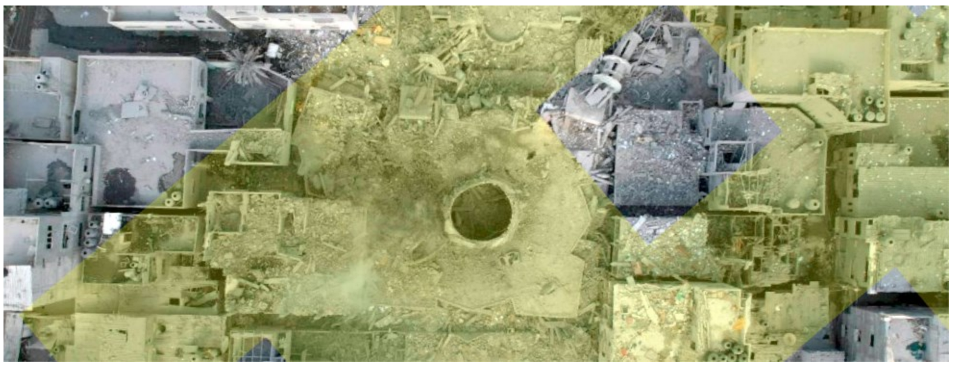
The row (b) represents changes related to craters. The New York Post (URL 16) suggests there are large quantities of craters from the same period. However, due to the limitations of available imagery, only craters outside of dense urban built-up areas are visible with PlanetScope. Therefore, the craters, and hence the images in row (b), are bordering the built-up mask. Even though IR-MAD was able to detect the change associated with one of the craters, the second one was outside the built-up mask generated from the dynamic world.

Row (c) is located up north in the AOI near the border with Israel, and according to the New York Post (URL 16) it's the enroute for tanks and heavy vehicles which would explain the significant changes, the area is also known to be full of Israeli fortifications, photographic evidence of bulldozers supports the fact that Israeli army bulldozes area including civil structures for military operation. Most abstractly, however the changes are again caused in uncovering of the soil and hence changing the SR values of the pixel. The difference is then high in comparison to the previous image showcasing change, especially if vegetation has been removed. The image in (c) is the location of the aggregated peaks of Z values in the figure 10. Interestingly, the clustering approach did not detect the changes in the structures, although they were directly affected.

Similar effect can be seen in the row (d) where urban vegetation was in form of gardens effectively wiped out, making it a significant change in both approaches and especially whilst utilizing the k-means clustering which precisely distinguished the structures from bare ground through the change layer, whereas the threshold suggests changes overreaching the bare soil and identified surrounding buildings as change. However, both in this case correctly identified change resembling complete annihilation of several structures.

In detecting changes related to rubble and building damage, the results showed disparity between the two approaches, in Figure 12, a city block with geolocated drone footage evidence sourced from CNN shows absolute destruction of the Ahmed Yassin Mosque in the area that occurred on the 9th of October (URL 17). Here, clustering did not detect a little to no changes whilst the threshold did. Showcasing that larger MAD variates values which were clustered together do not necessarily denote the qualitative side of change. Bombardment caused destruction is definitely a significant change associated with war related damage, but it does not necessarily correlate with higher MAD values.

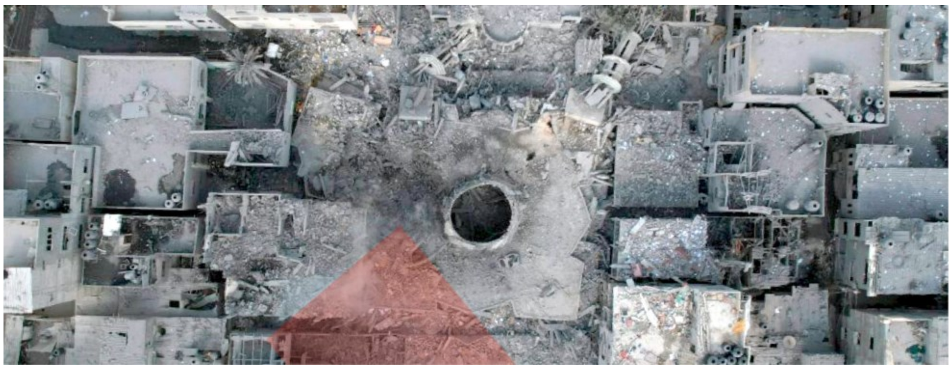
Figure 12: Georeferenced drone footage of Ahmed Yassin mosque post-airstrike (October 9) with 20-meter threshold change mask overlay



Source: CNN, author's work

The georeferencing was possible with the use of remaining edges of the surrounding buildings. In Figure 11 the mixed pixel problem is apparent, as the 20-m pixel can encompass 2 to 3 buildings, essentially combining the values of the heterogeneous surface. On the other hand, with a 20-m clustering approach with 71% OA, this approach was not able to correctly distinguish the changes and the destruction of this particular mosque except in part of the image (Figure 13).

Figure 13: Georeferenced drone footage of Ahmed Yassin mosque post-airstrike (October 9) with 20-meter cluster-based change mask overlay



Source: CNN, Author's work

Based on the results, a web application was created using the Google Earth Engine (GEE) platform to illustrate the temporal changes in the Gaza Strip throughout the conflict period.

The Iteratively Reweighted Multivariate Alteration Detection (IR-MAD) algorithm was applied to these dates and across the entire temporal range, from September 2023 to July 2024. Binary layers were exported using the superior threshold approach and visualized for each pair of dates. These visualizations are accessible through a GEE app at:

https://ee-cernikjac.projects.earthengine.app/view/change-detection-in-the-gaza-strip

which is embedded with other information about the research at:

https://sites.google.com/view/change-detection-gaza/home

8. Discussion and Conclusion

This research aimed to achieve three main goals: to utilize and evaluate the Iteratively Reweighted Multivariate Alteration Detection (IR-MAD) algorithm with Sentinel-2 data for CD in urban areas affected by war, to create Python scripts for CD analyses, and to develop a website with a web mapping application for publishing the results.

The IR-MAD method demonstrated promising results in detecting changes and war-related alterations, such as debris, destroyed houses, vegetation, and craters. A threshold derived from the significance level for the chi-square distribution proved superior to k-means clustering, achieving a 74% OA and identifying a 52% change between September 27th and November 26th, 2023. Resulting OA is 6% lower accuracy than the approach utilized by O. Bellinger (2023) whose SAR based workflow achieved 80% accuracy of detecting damaged buildings in Gaza. It is however important to note that the available resolution did not allow for assessment of buildings and IR-MAD was utilized to detect all changes, in that regard, visual inspection showed that IR-MAD was able to successfully attribute large values to the uncovered soil and ground related changes such as fortifications. Interestingly, 20-meter pixel sizes yielded better results than 10-meter pixels, which may be attributed to the dense urban built-up areas and the widespread effects of warfare, such as dust and debris. The mixed pixel problem or "mixels" could've been an advantage, as it scaled down the heterogeneity of the urban surface, helping to create a more integrated and consistent array of change pixels. Comparing the results of different pixel scales in settings with less substantial changes could, therefore, be valuable. In such scenarios, 10-meter or higher resolution data might be superior for detecting craters, changes in individual buildings, or convoys, as it allows for the detection of more isolated changes. These changes could be lost in the larger pixel size of lower-resolution datasets. An interesting implication of higher resolution imagery for IR-MAD could be in Ukraine where the scale of operations is so large and widespread that changes are more isolated.

Furthermore, It is important to note that the level of significance isn't a universal thresholding method and is partly specific to IR-MAD. Image *Z* follows a unimodal distribution, making it harder to threshold using conventional methods such as Otsu's algorithm. K-means clustering was therefore utilized as a potential way to create consistent binary maps. To enhance the results, thresholding with 6 classes was tested, and although it showed promising results for 20-meter classes, it lost its simplicity due to the need to combine classes into changes and no changes, and the resulting OA was lower at 71%.

Therefore, to save computational and time costs and the accuracy, the threshold-oriented method was chosen for binary classification in the web map, displaying change information through change and no-change masks across the whole Gaza Strip since the start of the conflict, exceeding the test AOI. The website was developed in Google Sites with the GEE app using the assets exported from the Python scripts.

Regarding the goal of developing the Python scripts, the goal was achieved, although most of the credit goes to the work of Dr. Mort Canty whose tutorials made it possible. His functions were turned into a Python library, for better utility, and custom functions such as masking using the Dynamic world were added together with an automatized process of exporting and thresholding GEE imagery, but the code still remains a derivative of the original tutorial. However, few visualization and testing scripts were also developed to display the results in 3D.

Although satisfactory results were achieved and IR-MAD can be utilized for monitoring changes in war affected urban areas; the methodology faced its limitations, particularly in the accuracy assessment. Although the randomly generated points were representative of the area, they may not have been representative of the research objective, meaning the points could have included an attribute for the ground truth type of change related to warfare (crater, strongpoint, etc). The quality of the points themselves is also in question, as PlanetScope could not achieve the most accurate reference data compared to sub-meter resolution imagery. Furthermore, thresholding, although effective, lacked a theoretical foundation comparable to other CD methods. A more robust thresholding method, perhaps based on machine learning hyper-optimization, could secure optimal thresholds consistently while utilizing a similar concept to this research. Additionally, masking the built-up area posed challenges. While damage to civil infrastructure is one of the most devastating effects of warfare, agriculture and natural areas are also affected by heavy machinery, fires, and chemicals.

In terms of tools and processing, the process of CD analysis also had its complications. Notably, failure to integrate accuracy assessment into python. After creation of the reference data in ArcGIS Pro, Accuracy Assessment was convenient in this environment, however, upon creating dozens of binary layers and confusion matrices, an automated python based accuracy function would not only save time but also help in future development for threshold optimization. Furthermore, regarding the GEE App, a more direct approach could be taken via the Streamlit option, making direct uploading and visualization straightforward and less path dependent.

However, these limitations can be improved upon. IR-MAD showed promising results in detecting changes in war-affected urban areas and hence opened the door for future improvement. Therefore, future developments would be:

- Developing a thresholding method that is consistent with across different images and yields higher accuracy
- The creation of a robust and contextually relevant reference layer for accuracy assessment, and this layer should subsequently be evaluated using IDE and not GIS in order to achieve better simplicity and workflow.
- Dual and separate analysis of urban areas and surrounding non-built up areas to further enhance context.
- Automatized framework for continuous update of GEE app binary layers or a Streamlit option directly from IDE.

9. References

Websites

[URL 1] GeeksforGeeks. ML | Fuzzy Clustering - GeeksforGeeks. [online]. [cit. 2024-05-05]. Available from: <https://www.geeksforgeeks.org/ml-fuzzy-clustering/>

[URL 2] ESA. Sentinel-2 operations. [online]. [cit. 2024-05-07]. Available from: <https://www.esa.int/Enabling_Support/Operations/Sentinel-2_operations>

[URL 3] Cenia. Land Cover Flows. [online]. [cit. 2024-06-11]. Available from: <https://landcover.cenia.cz/corine-land-cover/land-cover-flows/).>

[URL 4] Global Mangrove Watch. [online]. [cit. 2024-06-11]. Available from: <[https://www.globalmangrovewatch.org/>

[URL 5] Global Forest Watch. Forest Monitoring, Land Use & Deforestation Trends. [online]. [cit. 2024-06-11]. Available from: <https://www.globalforestwatch.org>

[URL 6] Atlas of Urban Expansion 2016, UNOHABITAT, NYU, Lincoln Institute of land policy. [online]. [cit. 2024-06-12]. Available from: <http://www.atlasofurbanexpansion.org>

[URL 7] Ballinger, O. A New Tool Allows Researchers to Track Damage in Gaza. Bellingcat. November 15, 2023. [online]. [cit. 2024-07-24]. Available from: <https://www.bellingcat.com/resources/2023/11/15/a-new-tool-allows-researchers-to-track-damage-in-gaza/>

[URL 8] UNRWA. UNRWA Situation Report #124 on the situation in the Gaza Strip and the West Bank, including East Jerusalem. June 24, 2024. [online]. [cit. 2024-07-28]. Available from: <https://www.unrwa.org/resources/reports/unrwa-situation-report-124-situation-gaza-strip-and-west-bank-including-east-Jerusalem>

[URL 9] State of Palestine - Subnational Administrative Boundaries - Humanitarian Data Exchange. [online]. [cit. 2024-05-02]. Available from: <https://data.humdata.org/dataset/cod-ab-pse?>

[URL 10] PlanetScope. [online]. [cit. 2024-06-07]. Available from: <https://developers.planet.com/docs/data/planetscope/>

[URL 11] Google Earth Engine. Google Earth Engine Tutorials, Python tutorials. [online]. [cit. 2024-01-20]. Available from: <https://developers.google.com/earth-engine/tutorials/community>

[URL 12] Canty, M. Change Detection in Google Earth Engine - The MAD Transformation (Part 2). Google for Developers. [online]. [cit. 2024-06-17]. Available from: <https://developers.google.com/earth-engine/tutorials/community/imad-tutorial-pt>

[URL 13] Omni Calculator. Critical value calculator. [online]. [cit. 2024-06-22]. Available from: <https://www.omnicalculator.com/statistics/critical-value>

[URL 14] Google Earth Engine. Unsupervised Classification (clustering). Google for Developers. [online]. [cit. 2024-06-27]. Available from: <https://developers.google.com/earth-engine/guides/clustering#colab-python>

[URL 15] Humanitarian Data Exchange. UNOSAT Gaza Strip Comprehensive Damage Assessment - 26 November 2023. [online]. [cit. 2024-07-11]. Available from: <https://data.humdata.org/dataset/unosat-gaza-strip-comprehensive-damage-assessment-26-november-2023?>

[URL 16] Erden, B., Levit, Z., Shao, E., Wallace, T., Boxerman, A. Maps: Tracking the Attacks in Israel and Gaza: Where Israeli forces are advancing toward Gaza. The New York Times. October 7, 2023. [online]. [cit. 2024-07-27]. Available from: <https://www.nytimes.com/interactive/2023/10/07/world/middleeast/israel-gaza-maps.html>

[URL 17] CNN Staff. Before and After Images show Gaza Mosque Devastation. CNN. October 18, 2023[online][cit. 2024-07-27]Available from: <**https://edition.cnn.com/2023/10/10/world/gaza-mosque-before-after-images-dg/index.html**>

Documents

AOUN, Joy-Fares; ARSHAD, Raja Rehan. 2017. *Syria Damage Assessment of Selected Cities: Aleppo, Hama, Idlib.* Washington, D.C.: World Bank Group. Available from: http://documents.worldbank.org/curated/en/530541512657033401/Syria-damage-assessment-of-selected-cities-Aleppo-Hama-Idlib.

AFAQ, Yasir and MANOCHA, Ankush, 2021. Analysis on change detection techniques for remote sensing applications: A review. Ecological Informatics. 1 July 2021. Vol. 63, p. 101310. DOI 10.1016/j.ecoinf.2021.101310.

AHMAD, Muhammad Nasar, SHAO, Zhenfeng and JAVED, Akib, 2023. Modelling land use/land cover (LULC) change dynamics, future prospects, and its environmental impacts based on geospatial data models and remote sensing data. Environmental Science and Pollution Research. 1 March 2023. Vol. 30, no. 12, p. 32985–33001. DOI 10.1007/s11356-022-24442-2.

ALCANTARA, Camilo, KUEMMERLE, Tobias, PRISHCHEPOV, Alexander V. and RADELOFF, Volker C., 2012. Mapping abandoned agriculture with multi-temporal MODIS satellite data. Remote Sensing of Environment. 1 September 2012. Vol. 124, p. 334–347. DOI 10.1016/j.rse.2012.05.019.

ASOKAN, Anju and ANITHA, J., 2019. Change detection techniques for remote sensing applications: a survey. Earth Science Informatics. June 2019. Vol. 12, no. 2, p. 143–160. DOI 10.1007/s12145-019-00380-5.

BAN, Yifang and YOUSIF, Osama, 2016. Change Detection Techniques: A Review. In: BAN, Yifang (ed.), Multitemporal Remote Sensing: Methods and Applications. Online.

Cham: Springer International Publishing. p. 19–43. ISBN 978-3-319-47037-5. [Accessed 1 July 2024].

BROWN, Christopher F., BRUMBY, Steven P., GUZDER-WILLIAMS, Brookie, BIRCH, Tanya, HYDE, Samantha Brooks, MAZZARIELLO, Joseph, CZERWINSKI, Wanda, PASQUARELLA, Valerie J., HAERTEL, Robert, ILYUSHCHENKO, Simon, SCHWEHR, Kurt, WEISSE, Mikaela, STOLLE, Fred, HANSON, Craig, GUINAN, Oliver, MOORE, Rebecca and TAIT, Alexander M., 2022. Dynamic World, Near real-time global 10 m land use land cover mapping. Scientific Data. 9 June 2022. Vol. 9, no. 1, p. 251. DOI 10.1038/s41597-022-01307-4.

CANTY, Morton John, 2019. *Image analysis, classification, and change detection in remote sensing: with algorithms for Python*. Fourth edition. Boca Raton: CRC Press, Taylor & Francis Group. ISBN 978-1-138-61322-5.

CHEN, Xi, LI, Jing, ZHANG, Yunfei, JIANG, Weiguo, TAO, Liangliang and SHEN, Wei, 2017. Evidential Fusion Based Technique for Detecting Landslide Barrier Lakes From Cloud-Covered Remote Sensing Images. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. May 2017. Vol. 10, no. 5, p. 1742–1757. DOI 10.1109/JSTARS.2017.2665529.

CHENG, Guangliang, HUANG, Yunmeng, LI, Xiangtai, LYU, Shuchang, XU, Zhaoyang, ZHAO, Qi and XIANG, Shiming, 2023. Change Detection Methods for Remote Sensing in the Last Decade: A Comprehensive Review.. Online. 9 May 2023. arXiv. arXiv:2305.05813. Available from: http://arxiv.org/abs/2305.05813 [Accessed 28 April 2024].

DEER, Peter, 1995. Digital Change Detection Techniques in Remote Sensing. Technical report (Electronics and Surveillance Research Laboratory (Australia)) ; DSTO-TR-0169.

DILIP, T., KUMARI, Mamta, MURTHY, C. S., NEELIMA, T. L., CHAKRABORTY, Abhishek and DEVI, M. Uma, 2023. Monitoring early-season agricultural drought using temporal Sentinel-1 SAR-based combined drought index. Environmental Monitoring and Assessment. 7 July 2023. Vol. 195, no. 8, p. 925. DOI 10.1007/s10661-023-11524-y.

DOXANI, G., KARANTZALOS, K. and STRATI, M. Tsakiri-, 2012. Monitoring urban changes based on scale-space filtering and object-oriented classification. International Journal of Applied Earth Observation and Geoinformation. 1 April 2012. Vol. 15, p. 38–48. DOI 10.1016/j.jag.2011.07.002.

FAKHRI, Falah and GKANATSIOS, Ioannis, 2021. Integration of Sentinel-1 and Sentinel-2 data for change detection: A case study in a war conflict area of Mosul city. *Remote Sensing Applications: Society and Environment*. April 2021. Vol. 22, p. 100505. DOI 10.1016/j.rsase.2021.100505.

EL-HATTAB, Mamdouh, 2016. Applying post classification change detection technique to monitor an Egyptian coastal zone (Abu Qir Bay). The Egyptian Journal of Remote Sensing and Space Science. 1 March 2016. Vol. 19. DOI 10.1016/j.ejrs.2016.02.002.

ESCWA, 2024.Assessment of physical damage caused to buildings by the war on Gaza: October 2023 – April 2024, Available from: assessment-physical-damage-buildings-war-gaza-english.pdf (unescwa.org)

GETU, Kenu and BHAT, H. Gangadhara, 2022. Dynamic simulation of urban growth and land use change using an integrated cellular automata and markov chain models: a case of Bahir Dar city, Ethiopia. Arabian Journal of Geosciences. 24 May 2022. Vol. 15, no. 11, p. 1049. DOI 10.1007/s12517-022-10304-1.

GHAZARYAN, Gohar, RIENOW, Andreas, OLDENBURG, Carsten, THONFELD, Frank, TRAMPNAU, Birte, STICKSEL, Sarah and JÜRGENS, Carsten, 2021. Monitoring of Urban Sprawl and Densification Processes in Western Germany in the Light of SDG Indicator 11.3.1 Based on an Automated Retrospective Classification Approach. Remote Sensing. 27 April 2021. Vol. 13, no. 9, p. 1694. DOI 10.3390/rs13091694.

HAMDI, Zayd Mahmoud, BRANDMEIER, Melanie and STRAUB, Christoph, 2019. Forest Damage Assessment Using Deep Learning on High Resolution Remote Sensing Data. Remote Sensing. January 2019. Vol. 11, no. 17, p. 1976. DOI 10.3390/rs11171976.

HANSEN, M. C., POTAPOV, P. V., MOORE, R., HANCHER, M., TURUBANOVA, S. A., TYUKAVINA, A., THAU, D., STEHMAN, S. V., GOETZ, S. J., LOVELAND, T. R., KOMMAREDDY, A., EGOROV, A., CHINI, L., JUSTICE, C. O. and TOWNSHEND, J. R. G., 2013. High-Resolution Global Maps of 21st-Century Forest Cover Change. Science. 15 November 2013. Vol. 342, no. 6160, p. 850–853. DOI 10.1126/science.1244693.

HANSEN, Matthew C, PICKENS, Amy and SONG, Zhen, [no date]. OPERA DIST product. University of Maryland, Department of Geographical Sciences, Global Land Analysis and Discovery (GLAD) laboratory.

HUESCA, Margarita, MERINO-DE-MIGUEL, Silvia, EKLUNDH, Lars, LITAGO, Javier, CICUÉNDEZ, Victor, RODRÍGUEZ-RASTRERO, Manuel, USTIN, Susan L. and PALACIOS-ORUETA, Alicia, 2015. Ecosystem functional assessment based on the "optical type" concept and self-similarity patterns: An application using MODIS-NDVI time series autocorrelation. International Journal of Applied Earth Observation and Geoinformation. 1 December 2015. Vol. 43, p. 132–148. DOI 10.1016/j.jag.2015.04.008.

HUSSAIN, Masroor, CHEN, Dongmei, CHENG, Angela, WEI, Hui and STANLEY, David, 2013. Change detection from remotely sensed images: From pixel-based to object-based approaches. ISPRS Journal of Photogrammetry and Remote Sensing. 1 June 2013. Vol. 80, p. 91–106. DOI 10.1016/j.isprsjprs.2013.03.006.

ISLAM, Kamrul, RAHMAN, Md. Farhadur and JASHIMUDDIN, Mohammed, 2018. Modeling land use change using Cellular Automata and Artificial Neural Network: The case of Chunati Wildlife Sanctuary, Bangladesh. Ecological Indicators. 1 May 2018. Vol. 88, p. 439–453. DOI 10.1016/j.ecolind.2018.01.047.

JENSEN, John R., 1983. Biophysical Remote Sensing. Annals of the Association of American Geographers. March 1983. Vol. 73, no. 1, p. 111–132. DOI 10.1111/j.1467-8306.1983.tb01399.x.

JOVANOVIĆ, Dušan, GAVRILOVIĆ, Milan, SLADIĆ, Dubravka, RADULOVIĆ, Aleksandra and GOVEDARICA, Miro, 2021. Building Change Detection Method to Support

Register of Identified Changes on Buildings. Remote Sensing. January 2021. Vol. 13, no. 16, p. 3150. DOI 10.3390/rs13163150.

KIM, Minhwa, PARK, Sang-Eun and LEE, Seung-Jae, 2023. Detection of Damaged Buildings Using Temporal SAR Data with Different Observation Modes. Remote Sensing. January 2023. Vol. 15, no. 2, p. 308. DOI 10.3390/rs15020308.

LA BARREDA-BAUTISTA, Betsabe De, A., Alejandra, COUTURIER, Stephane and LUIS, Jose, 2011. Tropical Dry Forests in the Global Picture: The Challenge of Remote Sensing-Based Change Detection in Tropical Dry Environments. In: CARAYANNIS, Elias (ed.), Planet Earth 2011 - Global Warming Challenges and Opportunities for Policy and Practice. Online. InTech. ISBN 978-953-307-733-8. [Accessed 31 March 2024].

Land Change Monitoring, Assessment, and Projection (LCMAP) Collection 1.0 Continuous Change Detection and Classification (CCDC) Algorithm Description Document (ADD), 2022. . Department of the Interior U.S. Geological Survey.

LI, Lu, WANG, Chao, ZHANG, Hong, ZHANG, Bo and WU, Fan, 2019. Urban Building Change Detection in SAR Images Using Combined Differential Image and Residual U-Net Network. Remote Sensing. January 2019. Vol. 11, no. 9, p. 1091. DOI 10.3390/rs11091091.

LIANG, Jiayong and LIU, Desheng, 2020. Estimating Daily Inundation Probability Using Remote Sensing, Riverine Flood, and Storm Surge Models: A Case of Hurricane Harvey. Remote Sensing. January 2020. Vol. 12, no. 9, p. 1495. DOI 10.3390/rs12091495.

LIU, Yang, SUN, Yujie, TAO, Shikang, WANG, Min, SHEN, Qian and HUANG, Jiru, 2021. Discovering Potential Illegal Construction Within Building Roofs from UAV Images Using Semantic Segmentation and Object-Based Change Detection. Photogrammetric Engineering & Remote Sensing. 1 April 2021. Vol. 87, no. 4, p. 263–271. DOI 10.14358/PERS.87.4.263.

LU, Dengsheng, MAUSEL, Paul, BRONDÍZIO, Eduardo and MORAN, Emilio, 2004.

Change Detection Techniques. International Journal of Remote Sensing. 1 January 2004. Vol. 25.

LUO, Hui, LIU, Chong, WU, Chen and GUO, Xian, 2018. Urban Change Detection Based on Dempster–Shafer Theory for Multitemporal Very High-Resolution Imagery. Remote Sensing. 21 June 2018. Vol. 10, no. 7, p. 980. DOI 10.3390/rs10070980.

MARX, Andrew J., 2016. Detecting urban destruction in Syria: A Landsat-based approach. *Remote Sensing Applications: Society and Environment*. October 2016. Vol. 4, p. 30–36. DOI 10.1016/j.rsase.2016.04.005.

MORA, Omar, LENZANO, M., TOTH, Charles, GREJNER-BRZEZINSKA, Dorota and FAYNE, Jessica, 2018. Landslide change detection based on Multi-Temporal airborne LIDAR-derived DEMs. Geosciences. 16 January 2018. Vol. 8, p. 23. DOI 10.3390/geosciences8010023.

MUELLER, Hannes, GROEGER, Andre, HERSH, Jonathan, MATRANGA, Andrea and SERRAT, Joan, 2021. Monitoring war destruction from space using machine learning. *Proceedings of the National Academy of Sciences*. 8 June 2021. Vol. 118, no. 23, p. e2025400118. DOI 10.1073/pnas.2025400118.

NIELSEN, Allan A., CONRADSEN, Knut and SIMPSON, James J., 1998. Multivariate Alteration Detection (MAD) and MAF Postprocessing in Multispectral, Bitemporal Image Data: New Approaches to Change Detection Studies. Remote Sensing of Environment. April 1998. Vol. 64, no. 1, p. 1–19. DOI 10.1016/S0034-4257(97)00162-

NIELSEN, Allan Aasbjerg, 2005. AN ITERATIVE EXTENSION TO THE MAD TRANSFORMATION FOR CHANGE DETECTION IN MULTI- AND HYPERSPECTRAL REMOTE SENSING DATA. . 2005.

PACHECO-PASCAGAZA, Ana María, GOU, Yaqing, LOUIS, Valentin, ROBERTS, John F., RODRÍGUEZ-VEIGA, Pedro, DA CONCEIÇÃO BISPO, Polyanna, ESPÍRITO-SANTO, Fernando D. B., ROBB, Ciaran, UPTON, Caroline, GALINDO, Gustavo, CABRERA, Edersson, PACHÓN CENDALES, Indira Paola, CASTILLO SANTIAGO, Miguel Angel, CARRILLO NEGRETE, Oswaldo, MENESES, Carmen, IÑIGUEZ, Marco and BALZTER, Heiko, 2022. Near Real-Time Change Detection System Using Sentinel-2 and Machine Learning: A Test for Mexican and Colombian Forests. Remote Sensing. January 2022. Vol. 14, no. 3, p. 707. DOI 10.3390/rs14030707.

PANUJU, Dyah, PAULL, David and GRIFFIN, Amy, 2020. Change Detection Techniques Based on Multispectral Images for Investigating Land Cover Dynamics. Remote Sensing. 1 June 2020. Vol. 12, p. 1781. DOI 10.3390/rs12111781.

PANUJU, Dyah R., PAULL, David J. and TRISASONGKO, Bambang H., 2019. Combining Binary and Post-Classification Change Analysis of Augmented ALOS Backscatter for Identifying Subtle Land Cover Changes. Remote Sensing. January 2019. Vol. 11, no. 1, p. 100. DOI 10.3390/rs11010100.

PARELIUS, Eleonora Jonasova, 2023. A Review of Deep-Learning Methods for Change Detection in Multispectral Remote Sensing Images. Remote Sensing. 16 April 2023. Vol. 15, no. 8, p. 2092. DOI 10.3390/rs15082092.

PENG, Daifeng and GUAN, Haiyan, 2019. Unsupervised change detection method based on saliency analysis and convolutional neural network. Journal of Applied Remote Sensing. 8 May 2019. Vol. 13, p. 1. DOI 10.1117/1.JRS.13.024512.

RAHMAN, Shoumik and MESEV, Victor, 2019. Change Vector Analysis, Tasseled Cap, and NDVI-NDMI for Measuring Land Use/Cover Changes Caused by a Sudden Short-Term Severe Drought: 2011 Texas Event. Remote Sensing. 23 September 2019. Vol. 11, p. 2217. DOI 10.3390/rs11192217.

REYMONDIN, Louis, JARVIS, Andrew, PEREZ-URIBE, Andres, TOUVAL, Jerry, ARGOTE, Karolina, COCA CASTRO, Alejandro, REBETEZ, Julien and GUEVARA, Edward, 2012. Terra-i A methodology for near real-time monitoring of habitat change at continental scales using MODIS-NDVI and TRMM.

SAMANTA, Sailesh, PAL, Dilip Kumar and PALSAMANTA, Babita, 2018. Flood susceptibility analysis through remote sensing, GIS and frequency ratio model. Applied Water Science. 21 April 2018. Vol. 8, no. 2, p. 66. DOI 10.1007/s13201-018-0710-1.

SEDIGHKIA, Mahdi and DATTA, Bithin, 2023. Detecting land use changes using hybrid machine learning methods in the Australian tropical regions. *GeoJournal*. 1 December 2023. Vol. 88, no. 1, p. 241–253. DOI 10.1007/s10708-022-10678-5.

SEYDI, Seyd Teymoor and HASANLOU, Mahdi, 2021. A New Structure for Binary and Multiple Hyperspectral Change Detection Based on Spectral Unmixing and Convolutional Neural Network. Measurement. 1 December 2021. Vol. 186, p. 110137. DOI 10.1016/j.measurement.2021.110137.

SINGH, Ashbindu, 1989. Review Article Digital change detection techniques using remotely-sensed data. *International Journal of Remote Sensing*. June 1989. Vol. 10, no. 6, p. 989–1003. DOI 10.1080/01431168908903939.

STILLA, Uwe and XU, Yusheng, 2023. Change detection of urban objects using 3D point clouds: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*. March 2023. Vol. 197, p. 228–255. DOI 10.1016/j.isprsjprs.2023.01.010.

WEST VIRGINIA UNIVERSITY and WARNER, Timothy, 2017. Nature of Multispectral Image Data. Geographic Information Science & Technology Body of Knowledge. Online. 17 July 2017. Vol. 2017, no. Q3. DOI 10.22224/gistbok/2017.3.1. [Accessed 28 June 2024].

YOU, Yanan, CAO, Jingyi and ZHOU, Wenli, 2020. A Survey of Change Detection Methods Based on Remote Sensing Images for Multi-Source and Multi-Objective Scenarios. Remote Sensing. January 2020. Vol. 12, no. 15, p. 2460. DOI 10.3390/rs12152460.

ZOVÁTHI, Örkény, NAGY, Balázs and BENEDEK, Csaba, 2022. Point cloud registration and change detection in urban environment using an onboard Lidar sensor and MLS reference data. International Journal of Applied Earth Observation and Geoinformation. 1 June 2022. Vol. 110, p. 102767. DOI 10.1016/j.jag.2022.102767.

Data sources

Copernicus (2023, 2024) Sentinel-2, Earth Engine Catalog [online]. Processed by ESA. Retrieved from Google Earth Engine Catalog.

Dynamic World (2023). Earth Engine Catalog [online]. Retrieved from Google Earth Engine Catalog.

Planet (2023). Planetscope [online]. Retrieved from: www.planet.com

UNOSAT (2023). Gaza Strip Comprehensive Damage Assessment – 26 November 2023 [online]. Retrieved from: https://data.humdata.org/dataset/unosat-gaza-strip-comprehensive-damage-assessment-26-november-2023?

OpenStreetMap contributors (2024). OpenStreetMap [online]. Retreived from: https://www.openstreetmap.org

ESRI (2023). Esri Landcover [online] Retreived from: https://livingatlas.arcgis.com/landcoverexplorer

10. Supplementary Materials

IR-MAD functions derived from Canty's tutorial

```python
#canty module for Google Earth Engine
import ee
"""
This module contains functions sourced from Dr. Mort Canty's tutorial on change detection
using the MAD transformation. For implementation. The module needs to be in the same
located in the same code as the code for executing the module.
The Tutorial is available at: https://developers.google.com/earth-
engine/tutorials/community/imad-tutorial-pt2
"""
ee.Authenticate()
# Initialize the library.
ee.Initialize(project='ee-thesiswar') # for intializing the project and acessing the API,
import geemap
import numpy as np
import random, time
import matplotlib.pyplot as plt
from scipy.stats import norm, chi2
from pprint import pprint  # for pretty printing
#########################################################
# MAD transformation
# Enter your own export to assets path name here -----------
EXPORT_PATH = 'projects/ee-thesiswar/assets/imad/'
print(EXPORT_PATH)
# ------------------------------------------------
def trunc(values, dec = 3):
    '''Truncate a 1-D array to dec decimal places.'''
    return np.trunc(values*10**dec)/(10**dec)
# Display an image in a one percent linear stretch.
def covarw(image, weights=None, scale=20, maxPixels=1e10):
    '''
    Return the centered image and its weighted covariance matrix.

    Parameters:
    - image: The input image.
    - weights: The weights to be used for the covariance calculation. If not provided, a
constant weight of 1 will be used.
    - scale: The scale at which to compute the covariance. Default is 20 scaling here
allowed for comparison of images with different resoltuons
    - maxPixels: The maximum number of pixels to compute the covariance. Default is 1e10.

    Returns:
    A tuple containing the centered image and its weighted covariance matrix.
    '''

    try:
        # Get geometry, band names, and number of bands.
        geometry = image.geometry()
        bandNames = image.bandNames()
        N = bandNames.length()

        # If weights are not provided, use a constant weight of 1.
        if weights is None:
            weights = image.constant(1)

        # Create an image with band names and weights.
        weightsImage = image.multiply(ee.Image.constant(0)).add(weights)

        # Compute means and centered image.
        means = image.addBands(weightsImage) \
                    .reduceRegion(ee.Reducer.mean().repeat(N).splitWeights(),
                                  scale=scale,
                                  maxPixels=maxPixels) \
                    .toArray() \
                    .project([1])
        centered = image.toArray().subtract(means)

        # Compute weighted covariance matrix.
        B1 = centered.bandNames().get(0)
        b1 = weights.bandNames().get(0)
        nPixels = ee.Number(centered.reduceRegion(ee.Reducer.count(),
                                                  scale=scale,
                                                  maxPixels=maxPixels).get(B1))
        sumWeights = ee.Number(weights.reduceRegion(ee.Reducer.sum(),
                                                      geometry=geometry,
                                                      scale=scale,
                                                      maxPixels=maxPixels).get(b1))
        covw = centered.multiply(weights.sqrt()) \
                    .toArray() \
                    .reduceRegion(ee.Reducer.centeredCovariance(),
                                      geometry=geometry,
                                      scale=scale,
                                      maxPixels=maxPixels) \
                    .get('array')
        covw = ee.Array(covw).multiply(nPixels).divide(sumWeights)

        return (centered.arrayFlatten([bandNames]), covw)

    except Exception as e:
        print('Error: %s' % e)
def corr(cov):
    '''
    Transfrom covariance matrix to correlation matrix.

    Parameters:
    - cov: The covariance matrix.

    Returns:
    The correlation matrix.
    '''

    # Diagonal matrix of inverse sigmas.
```

```python
    sInv = cov.matrixDiagonal().sqrt().matrixToDiag().matrixInverse()

    # Transform.
    corr = sInv.matrixMultiply(cov).matrixMultiply(sInv).getInfo()

    # Truncate.
    return [list(map(trunc, corr[i])) for i in range(len(corr))]

def geneiv(C, B):
    '''
    Return the eigenvalues and eigenvectors of the generalized eigenproblem

    Parameters:
    - C: A 2D array representing the matrix C.
    - B: A 2D array representing the matrix B.

    Returns:
    - A tuple (lambdas, eigenvecs) containing the eigenvalues and eigenvectors.
    Raises:
    - Any exception that occurs during the computation.

    Example usage:
    C = [[1, 2], [3, 4]]
    B = [[5, 6], [7, 8]]
    lambdas, eigenvecs = geneiv(C, B)
    print("Eigenvalues:", lambdas)
    print("Eigenvectors:", eigenvecs)
    '''

    try:
        # Convert input arrays to Earth Engine Arrays.
        C = ee.Array(C)
        B = ee.Array(B)

        # Compute the inverse of the Cholesky decomposition of B (Li = choldc(B)^-1).
        Li = ee.Array(B.matrixCholeskyDecomposition().get('L')).matrixInverse()

        # Solve the symmetric, ordinary eigenproblem Li*C*Li^T*x = lambda*x.
        Xa = Li.matrixMultiply(C) \
            .matrixMultiply(Li.matrixTranspose()) \
            .eigen()

        # Extract the eigenvalues as a row vector.
        lambdas = Xa.slice(1, 0, 1).matrixTranspose()

        # Extract the eigenvectors as columns.
        X = Xa.slice(1, 1).matrixTranspose()

        # Compute the generalized eigenvectors as columns by multiplying Li^T with X.
        eigenvecs = Li.matrixTranspose().matrixMultiply(X)

        # Return the eigenvalues and eigenvectors as a tuple.
        return (lambdas, eigenvecs)
    except Exception as e:
        print('Error: %s' % e)
def mad_run(image1, image2, scale=20):
    '''
    The MAD transformation of two multiband images.

    Parameters:
    - image1: The first multiband image.
    - image2: The second multiband image.
    - scale: The scale at which to compute the covariance. Default is 20.

    Returns:
    A tuple containing the U, V, MAD, and Z images
Raises:
    - Any exception that occurs during the computation.

    '''
    try:
        # Combine two images into one
        image = image1.addBands(image2)
        # Get the number of bands.
        nBands = image.bandNames().length().divide(2)
        # Compute the centered image and its weighted covariance matrix.
        centeredImage, covarArray = covarw(image, scale=scale)
        # Extract band names for the two sets of bands.
        bNames = centeredImage.bandNames()
        bNames1 = bNames.slice(0, nBands)
        bNames2 = bNames.slice(nBands)
        # Select bands for the two centered images.
        centeredImage1 = centeredImage.select(bNames1)
        centeredImage2 = centeredImage.select(bNames2)
        s11 = covarArray.slice(0, 0, nBands).slice(1, 0, nBands)
        s22 = covarArray.slice(0, nBands).slice(1, nBands)
        s12 = covarArray.slice(0, 0, nBands).slice(1, nBands)
        s21 = covarArray.slice(0, nBands).slice(1, 0, nBands)
        # Calculate matrices for generalized eigenproblems.
        c1 = s12.matrixMultiply(s22.matrixInverse()).matrixMultiply(s21)
        b1 = s11
        c2 = s21.matrixMultiply(s11.matrixInverse()).matrixMultiply(s12)
        b2 = s22
        # Solution of generalized eigenproblems.
        lambdas, A = geneiv(c1, b1)
        _, B = geneiv(c2, b2)
        rhos = lambdas.sqrt().project(ee.List([1]))
        # MAD variances.
        sigma2s = rhos.subtract(1).multiply(-2).toList()
        sigma2s = ee.Image.constant(sigma2s)
        # Ensure sum of positive correlations between X and U is positive.
        tmp = s11.matrixDiagonal().sqrt()
        ones = tmp.multiply(0).add(1)
        tmp = ones.divide(tmp).matrixToDiag()
        s = tmp.matrixMultiply(s11).matrixMultiply(A).reduce(ee.Reducer.sum(),
[0]).transpose()
        A = A.matrixMultiply(s.divide(s.abs()).matrixToDiag())
        # Ensure positive correlation.
        tmp = A.transpose().matrixMultiply(s12).matrixMultiply(B).matrixDiagonal()
        tmp = tmp.divide(tmp.abs()).matrixToDiag()
        B = B.matrixMultiply(tmp)
        # Canonical and MAD variates as images.
        centeredImage1Array = centeredImage1.toArray().toArray(1)
        centeredImage2Array = centeredImage2.toArray().toArray(1)
        U = ee.Image(A.transpose()).matrixMultiply(centeredImage1Array) \
                    .arrayProject([0]) \
                    .arrayFlatten([bNames2])
        V = ee.Image(B.transpose()).matrixMultiply(centeredImage2Array) \
                    .arrayProject([0]) \
                    .arrayFlatten([bNames2])
        MAD = U.subtract(V)
        # Chi-square image.
        Z = MAD.pow(2) \
               .divide(sigma2s) \
               .reduce(ee.Reducer.sum())
        return (U, V, MAD, Z)
    except Exception as e:
        print('Error: %s' % e)

 #########################################
# iMAD functions
def chi2cdf(Z, df):
        """
        Calculate the cumulative distribution function (CDF) of the chi-square
distribution.

        Parameters:
        Z (ee.Image): The input image representing the chi-square random variable. The
sum of squared Standardized MAD variates.
        df (int): The degrees of freedom of the chi-square distribution.Number of Bands -
1
        Returns:
        ee.Image: The image representing the CDF of the chi-square distribution.

        Notes:
        - The chi-square distribution is a continuous probability distribution that
arises in the context of
            hypothesis testing and confidence interval estimation for the variance of a
normally distributed population.
        - The CDF of the chi-square distribution gives the probability that a chi-square
random variable is less than or equal to a given value.

        """
        return ee.Image(Z.divide(2)).gammainc(ee.Number(df).divide(2))
def imad(current,prev):
    '''
    Iterator function for iMAD.
    Parameters:
    - current: The current iteration value.
    - prev: The previous iteration valu
    - returns done
    '''
    done =  ee.Number(ee.Dictionary(prev).get('done'))
    return ee.Algorithms.If(done, prev, imad1(current, prev))

def imad1(current,prev):
    '''
    Iteratively re-weighted MAD.

    Parameters:
    - current: The current iteration value.
    - prev: The previous iteration value.

    Returns:
    The updated iteration value.
    '''
    image = ee.Image(ee.Dictionary(prev).get('image'))
    Z = ee.Image(ee.Dictionary(prev).get('Z'))
    allrhos = ee.List(ee.Dictionary(prev).get('allrhos'))
    nBands = image.bandNames().length().divide(2)
    weights = chi2cdf(Z,nBands).subtract(1).multiply(-1)
    scale = ee.Dictionary(prev).getNumber('scale')
    niter = ee.Dictionary(prev).getNumber('niter')
    # Weighted stacked image and weighted covariance matrix.
    centeredImage, covarArray = covarw(image, weights, scale)
    bNames = centeredImage.bandNames()
    bNames1 = bNames.slice(0, nBands)
    bNames2 = bNames.slice(nBands)
    centeredImage1 = centeredImage.select(bNames1)
    centeredImage2 = centeredImage.select(bNames2)
    s11 = covarArray.slice(0, 0, nBands).slice(1, 0, nBands)
    s22 = covarArray.slice(0, nBands).slice(1, nBands)
    s12 = covarArray.slice(0, 0, nBands).slice(1, nBands)
    s21 = covarArray.slice(0, nBands).slice(1, 0, nBands)
    c1 = s12.matrixMultiply(s22.matrixInverse()).matrixMultiply(s21)
    b1 = s11
    c2 = s21.matrixMultiply(s11.matrixInverse()).matrixMultiply(s12)
    b2 = s22
    # Solution of generalized eigenproblems.
    lambdas, A = geneiv(c1, b1)
    _, B       = geneiv(c2, b2)
    rhos = lambdas.sqrt().project(ee.List([1]))
    # Test for convergence.
    lastrhos = ee.Array(allrhos.get(-1))
    done = rhos.subtract(lastrhos) \
               .abs() \
               .reduce(ee.Reducer.max(), ee.List([0])) \
               .lt(ee.Number(0.0001)) \
               .toList() \
               .get(0)
    allrhos = allrhos.cat([rhos.toList()])
    # MAD variances.
    sigma2s = rhos.subtract(1).multiply(-2).toList()
    sigma2s = ee.Image.constant(sigma2s)
    # Ensure sum of positive correlations between X and U is positive.
    tmp = s11.matrixDiagonal().sqrt()
    ones = tmp.multiply(0).add(1)
    tmp = ones.divide(tmp).matrixToDiag()
    s = tmp.matrixMultiply(s11).matrixMultiply(A).reduce(ee.Reducer.sum(),
[0]).transpose()
    A = A.matrixMultiply(s.divide(s.abs()).matrixToDiag())
    # Ensure positive correlation.
    tmp = A.transpose().matrixMultiply(s12).matrixMultiply(B).matrixDiagonal()
    tmp = tmp.divide(tmp.abs()).matrixToDiag()
    B = B.matrixMultiply(tmp)
    # Canonical and MAD variates.
    centeredImage1Array = centeredImage1.toArray().toArray(1)
    centeredImage2Array = centeredImage2.toArray().toArray(1)
    U = ee.Image(A.transpose()).matrixMultiply(centeredImage1Array) \
                   .arrayProject([0]) \
                   .arrayFlatten([bNames1])
    V = ee.Image(B.transpose()).matrixMultiply(centeredImage2Array) \
                   .arrayProject([0]) \
                   .arrayFlatten([bNames2])
    iMAD = U.subtract(V)
    # Chi-square image.
    Z = iMAD.pow(2) \
              .divide(sigma2s) \
              .reduce(ee.Reducer.sum())
    return ee.Dictionary({'done': done, 'scale': scale, 'niter': niter.add(1),
                          'image': image, 'allrhos': allrhos, 'Z': Z, 'iMAD': iMAD})

##########################################################
# to run imad
def run_imad(aoi, image1, image2, assetFN, scale=20, maxiter=100,):
    """
    Run the iMAD algorithm on two input images.

    Parameters:
    - aoi: Area of interest (ee.Geometry) to clip the output image.
    - image1: First input image (ee.Image).
    - image2: Second input image (ee.Image).
    - assetFN: Filename for exporting the iMAD result as an asset (str).
    - scale: Scale for the analysis (int, default=20).
    - maxiter: Maximum number of iterations for the iMAD algorithm (int,
default=100).Elsewise, the algorithm will stop when the change in the MAD variances is
less than 0.0001.
    """
    try:
        # Get the number of bands in the first input image
        N = image1.bandNames().length().getInfo()
        # Create a list of names for the iMAD images and the Z image
        imadnames = ['iMAD'+str(i+1) for i in range(N)]
        imadnames.append('Z')
        # Create a list of numbers from 1 to maxiter for iteration
        inputlist = ee.List.sequence(1, maxiter)
        # Create the initial dictionary for the first iteration
        first = ee.Dictionary({'done':0,
                    'scale': scale,
                    'niter': ee.Number(0),
                    'image': image1.addBands(image2),
                    'allrhos': [ee.List.sequence(1, N)],
                    'Z': ee.Image.constant(0),
                    'iMAD': ee.Image.constant(0)})
        # Iterate through the list of numbers using the imad function
        result = ee.Dictionary(inputlist.iterate(imad, first))
        # Retrieve the results from the iteration
        iMAD = ee.Image(result.get('iMAD')).clip(aoi)
        rhos = ee.String.encodeJSON(ee.List(result.get('allrhos')).get(-1))
        Z = ee.Image(result.get('Z'))
        niter = result.getNumber('niter')
        # Create an iMAD export image with the iMAD and Z bands
        iMAD_export = ee.Image.cat(iMAD, Z).rename(imadnames).set('rhos', rhos, 'niter',
niter)
        # Export the iMAD image to an asset
        assetId = EXPORT_PATH + assetFN
        assexport = ee.batch.Export.image.toAsset(iMAD_export,
                    description='assetExportTask',
                    assetId=assetId, scale=scale, maxPixels=1e10)
        assexport.start()
        # Print the export information
        print('Exporting iMAD to %s\n task id: %s'%(assetId, str(assexport.id)))
    except Exception as e:
        print('Error: %s'%e)
```

Main code executing Canty’s functions and handling data

```
# Project path for exporting-----------
EXPORT_PATH = 'projects/ee-thesiswar/assets/imad/'
print(EXPORT_PATH)
# ------------------------------------------------
import ee

# Trigger the authentication flow.
ee.Authenticate()

# Initialize the library.
ee.Initialize(project='ee-thesiswar')
```

```
print("hello world")
# Import other packages used in the tutorial
import canty # IRMAD>>this module needs to be saved in the same folder as the Jupyter
import geemap
import numpy as np
#####################################################################################
# Functions for handling data and masking
def collect(aoi, date, date2): # Collects the first image within the specified time range
for the first and second period, filters by aoi
    try:
        # Collect the first image within the specified time range for the first period
        im1 = ee.Image( ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")
                               .filterBounds(aoi)
                               .filterDate(ee.Date(date), ee.Date(date).advance(1,
'day'))

.filter(ee.Filter.contains(rightValue=aoi,leftField='.geo'))
                               .sort('CLOUDY_PIXEL_PERCENTAGE')
                               .first()
                               .clip(aoi) )

        # Collect the first image within the specified time range for the second period
        im2 = ee.Image( ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")
                               .filterBounds(aoi)
                               .filterDate(ee.Date(date2), ee.Date(date2).advance(1,
'day'))

.filter(ee.Filter.contains(rightValue=aoi,leftField='.geo'))
                               .sort('CLOUDY_PIXEL_PERCENTAGE')
                               .first()
                               .clip(aoi) )
        # Get the timestamps of the collected images
        timestamp1 = im1.date().format('E MMM dd HH:mm:ss YYYY')
        print(timestamp1.getInfo())
        timestamp2 = im2.date().format('E MMM dd HH:mm:ss YYYY')
        print(timestamp2.getInfo())
        # Get the image IDs
        image1_id = im1.id().getInfo()
        image2_id = im2.id().getInfo()

        # Print the image IDs
        print("Image 1 ID:", image1_id)
        print("Image 2 ID:", image2_id)

        # Return the collected images
        return (im1, im2)

    except Exception as e:
        print('Error: %s'%e)
def apply_mask(image, mask):
        """
        Apply a mask to a Sentinel image.
        Args:
            image (ee.Image): Sentinel image.
            mask (ee.Image): Mask image.

        Returns:
            ee.Image: Sentinel image with the mask applied.
        """
        masked_image = image.updateMask(mask)

        return masked_image
    # Load ESA WorldCover dataset
def export_image(image, folder_path, image_name, scale=20):
    """
    Export an image from Google Earth Engine using geemap.

    Args:
        image (ee.image.Image): The image to export.
        folder_path (str): The path to the folder where the image will be exported.
        image_name (str): The name to use for the exported file.
        scale (int, optional): The scale of the exported image. Defaults to 30.
    """
    # Create the full export path
    export_path = f"{folder_path}\\{image_name}.tif"

    # Create a geemap Map instance
    Map = geemap.Map()

    # Add the image to the map
    Map.addLayer(image, {}, 'Image')

    # Export the image
    geemap.ee_export_image(image, export_path, scale=scale)

    # Display the map
    Map
def process_images(aoi, dates, visirbands, city_mask):
    """
    Processes images based on the provided area of interest (AOI), dates,
visible/infrared bands, and city mask.

    Parameters:
    aoi (ee.Geometry): The area of interest for image processing.
    dates (list): A list of date strings in the format 'YYYY-MM-DD' to filter the image
collection.
    visirbands (list): A list of band names to select from the image collection.
    city_mask (ee.Image): An image mask to apply to the processed images.

    Returns:
    None
    """

    for i in range(len(dates) - 1):
        date1 = dates[i]
        date2 = dates[i + 1]

        # Collect the two Sentinel-2 images for the specified dates
        im1, im2 = collect(aoi, date1, date2)

        # Apply the city mask to the images
        im1 = apply_mask(im1, city_mask)
        im2 = apply_mask(im2, city_mask)

        # Generate the output name based on the date range
        output_name = f'FINAL20MAcrosstherange_{date1}_{date2}'
        file_names.append(output_name)  # Append the output name to the list

        # Run the IMAD function on the masked images
        canty.run_imad(aoi, im1.select(visirbands), im2.select(visirbands), output_name)

# Folder path for exporting images to the local machine
folder_path = r'C:\Users\jachy\Desktop\iMAD\Outputs'
######################################################################################
# Areo of interest (AOI) for the study as a featureCollection derived from a shapefile
located as an asset in the Earth Engine
aoi = ee.FeatureCollection(
    'projects/ee-thesiswar/assets/Gaza_AOI').geometry() #North Gaza
######################################################################################
# Masking using the Dynamic World
START = ee.Date('2023-09-27') # select desired day
# Define the end date by advancing the start date by one day
END = START.advance(1, 'day')

# Create a composite filter that combines spatial and temporal filters
col_filter = ee.Filter.And(
    ee.Filter.bounds(ee.Geometry(aoi)),  # Filter to include only images intersecting the
area of interest (AOI)
    ee.Filter.date(START, END),  # Filter to include only images within the specified
date range
)

# Apply the filter to the Dynamic World Image Collection
dw_col = ee.ImageCollection('GOOGLE/DYNAMICWORLD/V1').filter(col_filter)

# Define the class names present in the Dynamic World dataset
CLASS_NAMES = [
    'water',
    'trees',
    'grass',
    'flooded_vegetation',
    'crops',
    'shrub_and_scrub',
    'built',
    'bare',
    'snow_and_ice',
]
```

```
# Extract the first image from the filtered collection
dw_image = ee.Image(dw_col.first())
# Clip the Dynamic World image to the area of interest
dw_image_clipped = dw_image.clip(aoi)
built_mask = dw_image_clipped.select('label').eq(CLASS_NAMES.index('built'))

# Generate a binary layer where 'built' areas are 1 and others are 0, then apply selfMask
to keep only 'built' areas
binary_layer = built_mask.gt(0).selfMask()
# Update the mask of the built_mask layer with the binary_layer to isolate 'built' areas
city_mask = built_mask.updateMask(binary_layer)
##############################################################################
# execution of Pre-procesing
# Define the bands to be used in the analysis
visirbands = ['B2', 'B3', 'B4', 'B8', 'B11', 'B12']
# Define the dates for the two periods
file_names = [] # empty list to save the names of the processed images
# Batch Processing of dates
dates = [
    "2023-09-27",
    "2023-11-01",
    "2023-11-26",
    "2023-12-26",
    "2024-01-20",
    "2024-03-05",
    "2024-04-04",
    "2024-05-09",
    "2024-06-18",
    "2024-07-18",
]
dates =  ["2023-09-27","2024-07-23"]# for individual processing, in case of not needing
batch
process_images(aoi, dates, visirbands, city_mask) # Function to process images in batch
and send tasks to GEE for processing
# Run the IRMAD >> parameters: (aoi, image1, image2, output_name) individdually
#canty.run_imad(aoi, im1.select(visirbands), im2.select(visirbands),'ExampleIRMAD') #
Function to run IRMAD on two images
########################################################################################
alpha_values = [0.00005]
# Iterate through all the p values in the list (option for more)
# Iterate through each file name.and export the binary masks locally into the a local
machine
for file_name in file_names:
    try:
        # Load the image from GEE.
        im_z = ee.Image(EXPORT_PATH + file_name).select(6).rename('Z') # assuming that
the 6th band is the Z-score and we have 6 bands for the analysis

        # p-values image by canty
        pval = canty.chi2cdf(im_z, 6).subtract(1).multiply(-1).rename('pval')
        # Iterate through all the p values in the list
        for p_value in alpha_values:
            # Create a binary mask: 1 where pval is less than p_value (indicating
change), 0 otherwise (indicating no change)
            binaryMask = pval.lt(p_value).rename('binaryMask')

            # Define the export name by removing the dot from the p_value
            export_name = f'MASKS{str(p_value).replace(".", "_")}_{file_name}'

            # Export the binaryMask with the unique name
            #export_image(binaryMask, folder_path, export_name, scale=20) # Export the
binary mask
            print(f"Exported mask with p-value: {p_value} for {file_name}")

    except Exception as e:
        print(f"Error processing file {file_name}: {e}")

print("Export is finished.")
# Iterate through each file name.for clustering
region = aoi
for file_name in file_names:
    try:
        # Load the image from GEE.
        input = ee.Image(EXPORT_PATH + file_name).select(0, 1, 2, 3, 4, 5)

        # Make the training dataset.and set scales
        training = input.sample(region=region, scale=20, numPixels=50000)

        # Instantiate the clusterer and train it.based on the training dataset and the
mean
        clusterer = ee.Clusterer.wekaKMeans(10).train(training)

        # Cluster the input using the trained clusterer.
        result = input.cluster(clusterer)
        # Export the clustered image with a unique name.
        export_name = f'cluster20_6classes10ss{file_name}'
        #export_image(result, folder_path, export_name, scale=20)
        print(f"Exported cluster for {file_name}")

    except Exception as e:
        print(f"Error processing file {file_name}: {e}")

print("Export is finished.")
```

Visualizing Z values in 3D

```
# Preliminaries such as AOI and previous modules + functions need to be in the same
jupyter notebook for this code to work
# the output shoudl visualize the image in 3D space based its values
import plotly.graph_objects as go
```

```python
import kaleido

#dataset = ee.Image(EXPORT_PATH + filename).select(6).rename('Z')

region = aoi
#10mVisirbandSS
# Get the elevation data within the region
elevation = dataset.select('Z').clip(region)

# Define a scale in meters
scale = 20

# Reduce the region to a numpy array
elevation_data = geemap.ee_to_numpy(elevation, region=region, scale=scale)

# Downsample if necessary to avoid performance issues
# Aggregate data into a lower resolution grid
def aggregate_data(data, factor):
    """Aggregate data by a factor, averaging over each block."""
    new_shape = (data.shape[0] // factor, data.shape[1] // factor)
    aggregated_data = np.zeros(new_shape)
    for i in range(new_shape[0]):
        for j in range(new_shape[1]):
            block = data[i*factor:(i+1)*factor, j*factor:(j+1)*factor]
            aggregated_data[i, j] = np.mean(block)
    return aggregated_data

# Set the aggregation factor (e.g., 5x5 pixels combined into one)
aggregation_factor = 5
elevation_data_aggregated = aggregate_data(elevation_data, aggregation_factor)

# Create x and y coordinates based on the aggregated data dimensions
nrows, ncols = elevation_data_aggregated.shape
x = np.linspace(0, ncols - 1, ncols)
y = np.linspace(0, nrows - 1, nrows)
x, y = np.meshgrid(x, y)

# Define a custom colorscale with white for zero values
colorscale = [
    [0, 'white'],       # White for zero values
    [0.001, 'white'],
    [0.01, 'blue'],      # Transition to blue for low values
    [0.1, 'yellow'],     # Green for mid-range values
    [1, 'red']          # Red for high values
]

# Generate 3D plot using plotly
fig = go.Figure(data=[go.Surface(z=elevation_data_aggregated, x=y, y=x,
colorscale=colorscale)])

# Update layout for better visualization
# Update layout to hide X and Y axes
fig.update_layout(
    scene=dict(
        xaxis=dict(
            title='',
            showticklabels=False,
            showbackground=False,
            zeroline=False  # Remove the pedestal
        ),
        yaxis=dict(
            title='',
            showticklabels=False,
            showbackground=False,
            zeroline=False  # Remove the pedestal
        ),
        zaxis=dict(
            title='Z values',
            titlefont=dict(size=18),  # Increase font size of Z axis title
            tickfont=dict(size=14)    # Increase font size of Z axis ticks
        ),
        camera=dict(
            eye=dict(x=1, y=0.2, z=1)  # Adjust these values to rotate the plot
        )
    ),
    legend=dict(
        orientation="h",  # Make the legend horizontal
        x=0.5,
        xanchor="center",
        y=-0.1,
        yanchor="top",
        font=dict(size=14)  # Increase font size of legend
    )
)

# Show the plot
#fig.show()
fig.show(renderer="png", width=2000, height=1500)
```