Assessing War Damage in Kharkiv and Mariupol, Ukraine Using Sentinel-1 Imagery

CASA0025- Group: SixQL

Tianqi CHU, Yujia MA, Yiyao CUI, Ruiya Lin, Siyuan FENG, Yunqian YAO

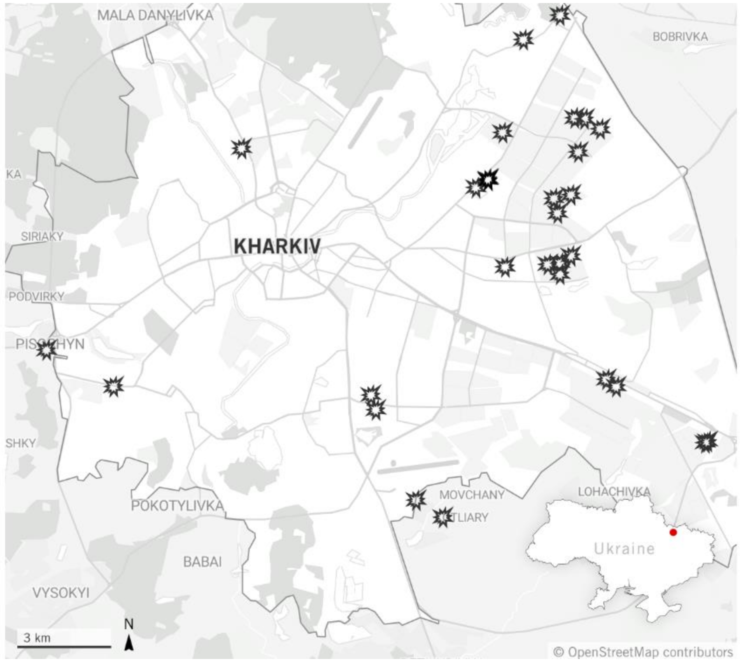
Project Foundations

Motivation | End Users | Data | Methodology Framework

Motivation

The Russia–Ukraine war has caused widespread urban destruction. We focus on Kharkiv and Mariupol, two of the hardest-hit cities.

- Existing tools are costly and limited
- Damage data is delayed or unavailable
- Recovery needs fast, interpretable estimates
- Our application uses open Sentinel-1 data for rapid analysis



Strikes in Kharkiv – June 2022, Ukraine War
Source: Amnesty International

End Users

Planning departments

Prioritise reconstruction based on damage to key facilities

Humanitarian organisations

Identify critical damage to guide shelter and aid delivery

Disaster assessment professionals

Use damage data for validation, change tracking, and comparison

Public & media

Explore visual maps to raise awareness and support reporting

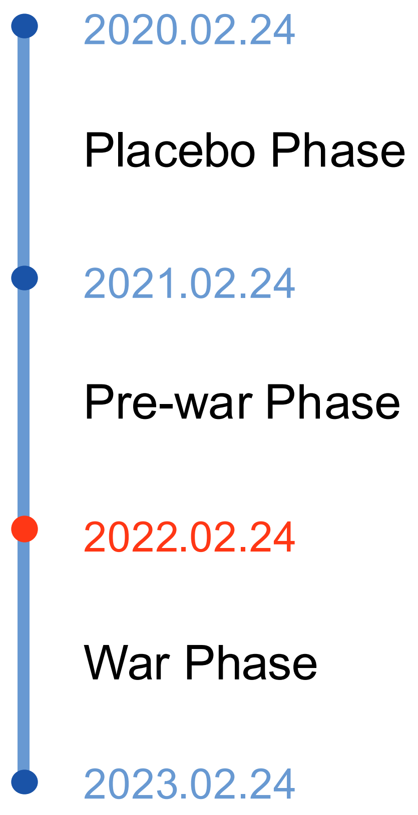
Data

■ Sentinel-1 Imagery
- Source: Google Earth Engine (2020.2.24–2023.2.24)
- VH polarization only

■ Building Footprints
- Source: Overture Maps Foundation
- Filtered to exclude buildings <50 m^2

■ Road Networks
- Source: OpenStreetMap
- trunk, primary, secondary, and tertiary roads

Sentinel-1 Imagery Time Phases



Methodology Framework

Data

Google Earth Engine
(filtering by GEE)
Sentinel-1 Imagery

Overture Maps
(processing by R)
Building Footprints

OpenStreetMap
(processing by python)
Road Networks

2020-02-24 to 2021-02-24 images

Methods

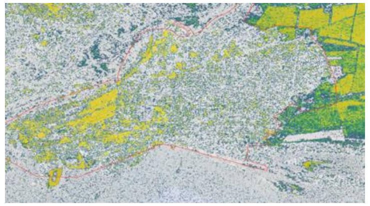
T-Test
(with t=2.5 ~99% confident interval)
*compare **monthly***
from 2022-02-24 to 2023-02-24

overlay on the t-test results

Placebo Test & Validation
compared the outcome to UNOSAT data

User Interface & Functions

Damage Proportion Distribution Map



Presenting the T-test detection results

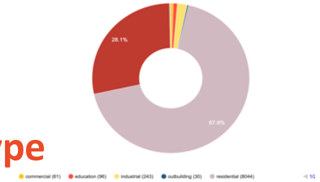
Damage Proportion (Building) Map

Damage Proportion (Road) Map

Finding out the locations of damaged buildings and roads, and where should be the major reconstruction areas

Damaged Building & Road Info

Percentages of Damaged Building by Subtype



Exploring the whole damage for each city and which types of buildings need more reconstruction

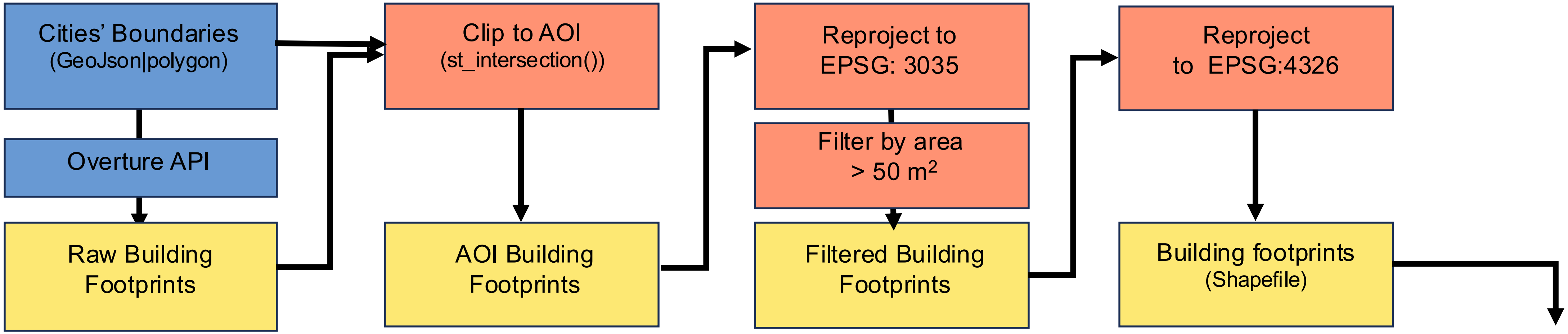
Application Link: https://ee-yiyaocui.projects.earthengine.app/view/ukraine-buildings-and-roads-damage-assessment

Methodology

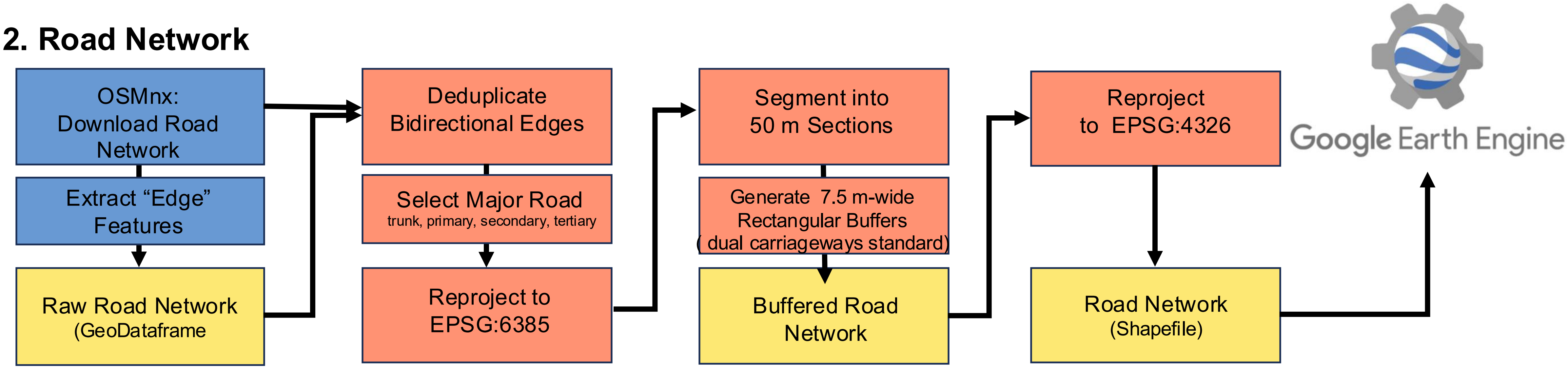
Data processing | T-test (GEE) | Validation & Limitation

Data Collection & Processing (Vector)

1. Building Footprints



2. Road Network



T-test (GEE)

t-test Function	Sentinel-1 Image Collection & Generation	Damage Calculation	Visualisation
Perform pixel-level t-tests, calculating radar backscatter changes and **identifying significant changes** between pre- and post-war images.	Filters Sentinel-1 imagery by polarization, orbit direction, and relative orbit number, applies an NDVI-based mask to remove vegetation, and **generates a composite change intensity map**.	Extract and **quantify damage levels** for building footprints and road networks using pixel-based change thresholds.	**Visualises the results** by displaying changes in damaged buildings and roads, with contours and boundary layers to highlight the damage.

t-test Function

input variable

s1: Sentinel-1 Image Collection; ***shock***: Event (War) Start Date;
pre-interval: Pre-event Time Window; ***post_interval***: Post-event Time Window

Initial time setting

Image Collection

Statistical Computation

t-test Computation

```
// ---------------------------------------- TTest function  -------------------------------
// t-test function (building analysis)
function ttest(s1, shock, pre_interval, post_interval) {
  // Convert event date to ee.Date object
  var shock = ee.Date(shock);
  // Filter pre-event images: from (shock - pre_interval months) to shock
  var pre = s1.filterDate(
    shock.advance(ee.Number(pre_interval).multiply(-1), "month"),
    shock
  );
  // Filter post-event images: from shock to (shock + post_interval months)
  var post = s1.filterDate(shock, shock.advance(post_interval, "month"));
  // Calculate pre-event mean, standard deviation, and image count
  var pre_mean = pre.mean();
  var pre_sd = pre.reduce(ee.Reducer.stdDev());
  var pre_n = ee.Number(pre.filterBounds(aoi).size());
  // Calculate post-event mean, standard deviation, and image count
  var post_mean = post.mean();
  var post_sd = post.reduce(ee.Reducer.stdDev());
  var post_n = ee.Number(post.filterBounds(aoi).size());
  // Print pre and post event image counts for debugging
  print('Pre-event images count: ', pre_n);
  print('Post-event images count: ', post_n);
  // Calculate pooled standard deviation
  var pooled_sd = pre_sd
    .multiply(pre_sd)
    .multiply(pre_n.subtract(1))
    .add(post_sd.multiply(post_sd).multiply(post_n.subtract(1)))
    .divide(pre_n.add(post_n).subtract(2))
    .sqrt();
  // Calculate denominator part of t-test formula
  var denom = pooled_sd.multiply(
    ee.Number(1).divide(pre_n).add(ee.Number(1).divide(post_n)).sqrt()
  );
  // Calculate degrees of freedom (number of observations minus 2)
  var df = pre_n.add(post_n).subtract(2);
  print("Number of Images: ", df);
  // Calculate t-value: absolute difference of means divided by denominator, minus 2
  var change = post_mean
    .abs()
    .subtract(pre_mean.abs())
    .divide(denom)
    .abs()
    .subtract(2.5);

  // Return t-value for each pixel
  return change;
}
```

intermediate variable

pre_mean: x_1; ***pre_sd:*** s_1^2; ***pre_n:*** n_1
post_mean: x_2; ***post_sd:*** s_2^2; ***post_n:*** n_2

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

denom: Denominator of the Formula

df: Degrees of Freedom

output variable

change: Absolute t-value minus 2, indicating significant spatial changes when > 0.

Sentinel-1 Image Collection & Generation

```
// Generic Sentinel-1 filtering function
function filter_s1(path) {
  var s1 = ee
    .ImageCollection("COPERNICUS/S1_GRD")
    .filter(ee.Filter.listContains("transmitterReceiverPolarisation", "VH"))
    .filter(ee.Filter.eq("instrumentMode", "IW"))
    .filter(ee.Filter.eq("orbitProperties_pass", path))
    .filterBounds(aoi)
    .filterDate(startDate, endDate)
    .select("VH");

  var orbit = s1
    .aggregate_array("relativeOrbitNumber_start")
    .reduce(ee.Reducer.mode());

  s1 = s1.filter(ee.Filter.eq("relativeOrbitNumber_start", orbit));

  // Select different t-test function based on analysis type
  var change = ttest(s1, SHOCK_DATE, PRE_INTERVAL, POST_INTERVAL)
  return change;
}
// ------------------------------------- Calculation  -----------------------
// Select sentinel image for calculation
var ascending_image = filter_s1("ASCENDING");
var descending_image = filter_s1("DESCENDING");

var asc_des = ee.ImageCollection([ascending_image, descending_image])
  .median()
  .clip(aoi);

// Add NDVI Mask
var s2 = ee.ImageCollection('COPERNICUS/S2_SR')
  .filterDate(startDate, SHOCK_DATE)
  .filterBounds(aoi)
  .select(['B4','B8'])
  .median();
var ndvi = s2.normalizedDifference(['B8','B4']).rename('NDVI');
var ndvi_mask = ndvi.gt(0.2);

// Generate Composite Image
var composite_image = asc_des.where(ndvi_mask, 0);
```

path: ASCENDING or DESCENDING

VH: VH polarization band, sensitive to surface structural changes

orbit: The most common relative orbit number

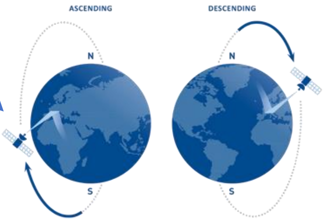
change: Output images based on t-test

ndvi_mask: Vegetation mask with NDVI greater than 0.2

composite_image: Final composite image after removing vegetation

Sentinel-1 Radar Image Filtering

Select the VH polarization band & the most common orbit number to ensure consistency.



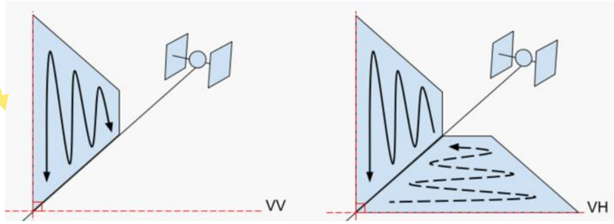


Image Composition (ASC & DES)

Extract ascending and descending orbit change maps separately and compose a median image, reducing orbit difference impacts on results.

NDVI Vegetation Mask Generation

Calculate the NDVI index to generate a vegetation mask in order to avoid vegetation-induced noise.

Image Source：https://forum.step.esa.int/t/ascending-descending/16946/11, https://hyp3-docs.asf.alaska.edu/guides/introduction_to_sar/

Damage Calculation

Damage Calculation of Buildings and Roads

threshold: Sets the t-test result threshold to select pixels identified as damaged

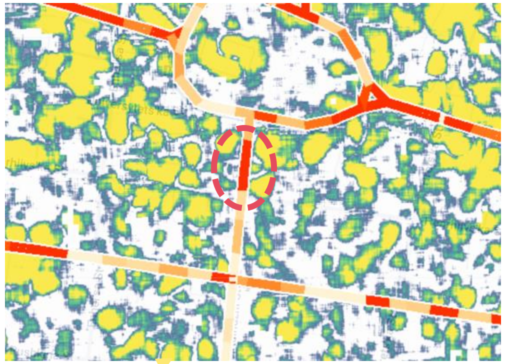
```
// select significant pixel
var threshold = composite_image.updateMask(composite_image.gt(0));

// damaged building detection
var damaged_building = threshold.reduceRegions({
  collection: buildings,
  reducer: ee.Reducer.mean(),
  scale: 10
});


// damaged road detection
var damaged_roads = threshold.reduceRegions({
  collection: roads,
  reducer: ee.Reducer.mean(),
  scale: 10
});

// -------------------------------------- Calculation --------------------------------------
var mariupol_buildings = damaged_building.filterBounds(ukr1).filter(ee.Filter.gt("mean", 0));
var kharkiv_buildings = damaged_building.filterBounds(ukr7).filter(ee.Filter.gt("mean", 0));
var total_buildings = mariupol_buildings.merge(kharkiv_buildings);

var mariupol_roads = damaged_roads.filterBounds(ukr1).filter(ee.Filter.gt("mean", 0.5));
var kharkiv_roads = damaged_roads.filterBounds(ukr7).filter(ee.Filter.gt("mean", 0.5));
var total_roads = mariupol_roads.merge(kharkiv_roads);
```

damaged_building: Mean pixel value for buildings

scale: Spatial resolution of the pixels, 10m

damaged_roads: Mean pixel value for roads

mariupol, kharkiv_buildings/roads: Mean values for buildings and roads in each city.



0.5: For roads, the threshold is set to 0.5, as road results are more sensitive to surrounding areas. This means a pixel must have over 50% significant values to be considered damaged.

Visualisation

```
// -------------------------------------- Visualization ---------------------------------
//ttest
Map.addLayer(
  composite_image,
  { min: 0, max: 4, opacity: 0.8, palette: palette },
  "Buildings Change",
  false  // Not displayed by default
);
```

Add t-test Result Layer

```
// Building Footprint Outline
var empty_buildings = ee.Image().byte();
var buildings_outline = empty_buildings.paint({
  featureCollection: damaged_building,
  color: "mean",
  width: 1.5
});

// Draw road outline - add white border
var empty_roads = ee.Image().byte();

// First create a wider white border
var roads_white_outline = empty_roads.paint({
  featureCollection: damaged_roads,
  color: 1,  // White
  width: 9   // Wider than normal width to form a border
});

// Then create normal width colored roads
var roads_outline = empty_roads.paint({
  featureCollection: damaged_roads,
  color: "mean",
  width: 5
});
```

Building, Road, Road Outline Layers

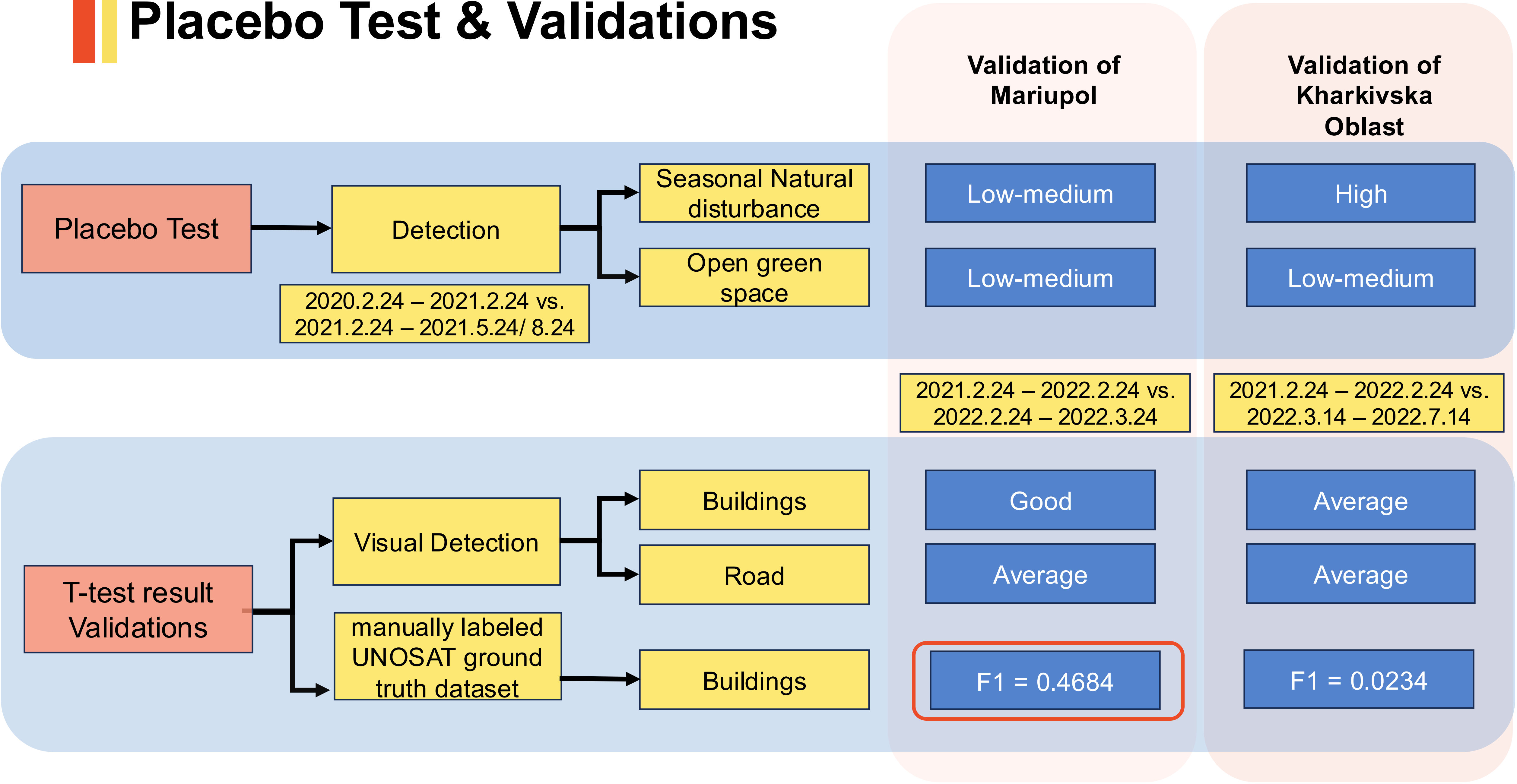
min, max, color, width: The parameters that control the display of the layers

```
// Add building outline layer
Map.addLayer(
  buildings_outline,
  { palette: building_palette, min: 0.6, max: 2 },
  "Damaged Buildings",
  true  // Displayed by default
);
```

Add Building and Road Layers onto the Map

```
// First add white border layer
Map.addLayer(
  roads_white_outline,
  { palette: ["ffffff"], min: 0, max: 1 },
  "Roads White Outline",
  false  // Not displayed by default
);

// Then add colored road layer
Map.addLayer(
  roads_outline,
  { palette: building_palette, min: 0.6, max: 2 },
  "Damaged Roads",
  false  // Not displayed by default
);

// Add city boundary layer
Map.addLayer(cityBoundaries, {}, 'adminboundaries');
```

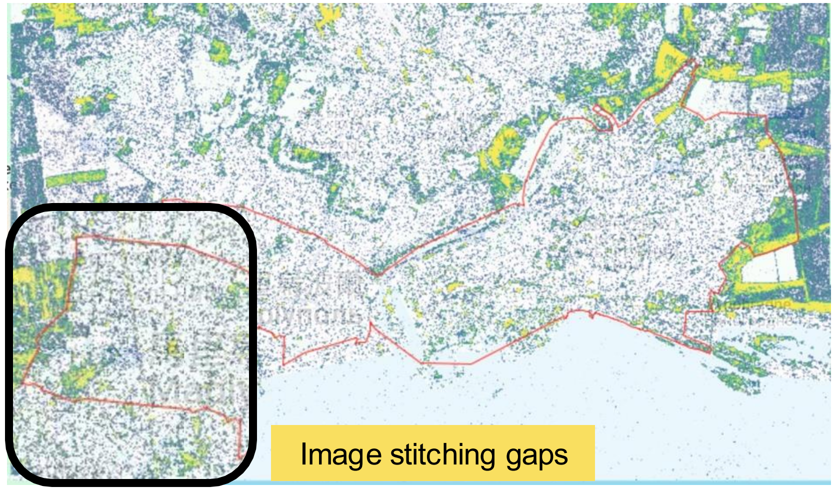
Placebo Test & Validations



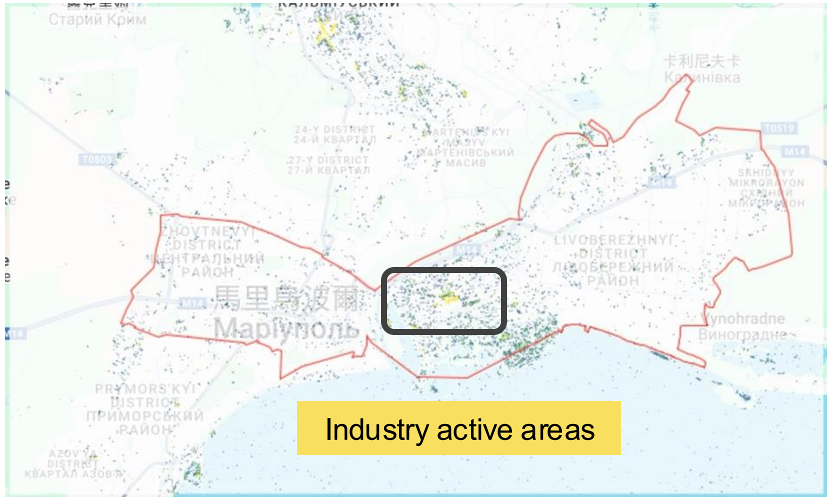
Placebo Test

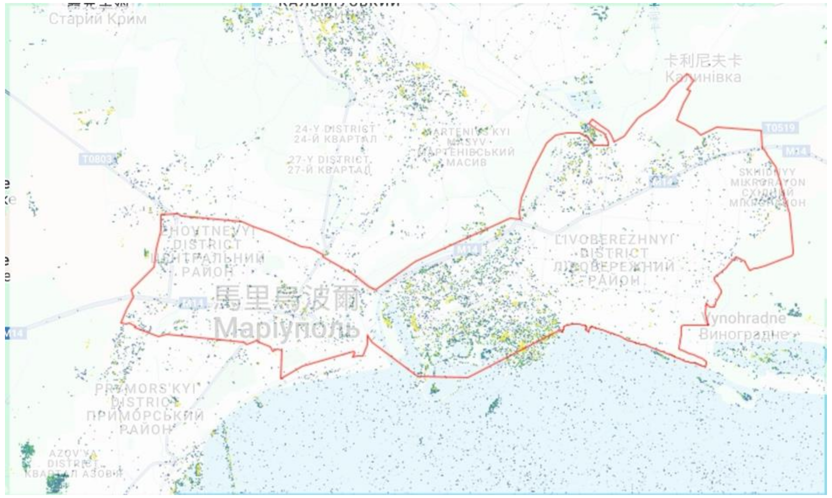
Mariupol

Before removal of vegetation impacts



After removal of vegetation impacts





3 months (until 2021.5.24)

9 months (until 2021.12.24)

Detection:

- The coastal city of Mariupol experienced less seasonal disturbance, especially after vegetation data was excluded.
- Satellite image stitching gaps appears.
- Disturbances were concentrated in the city center, particularly in industrially active areas such as engineering zones.

Placebo Test

Kharkivska Oblast

Before removal of vegetation impacts

After removal of vegetation impacts

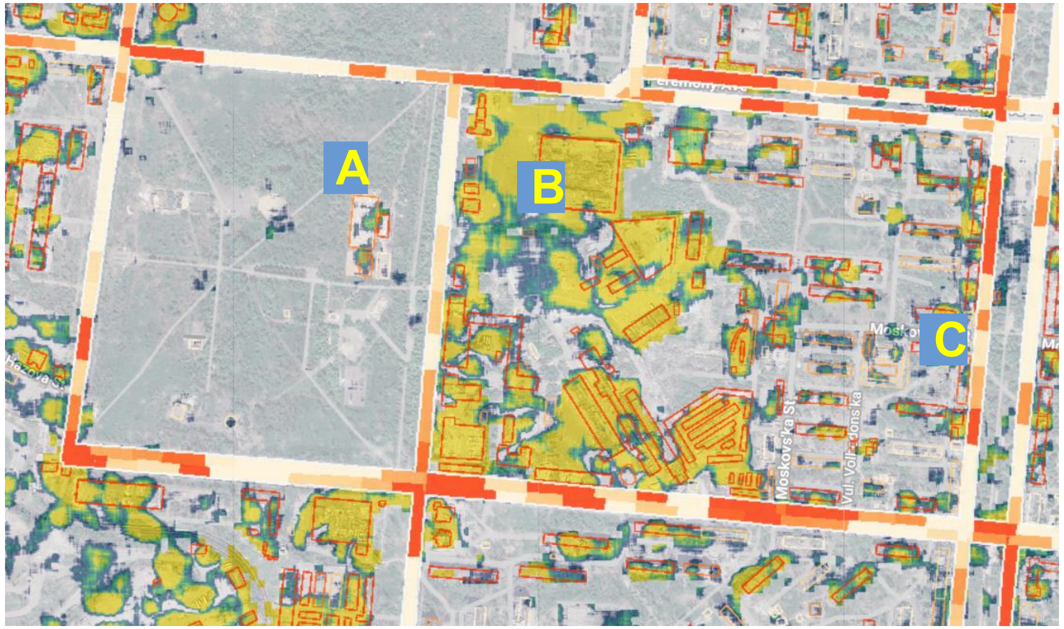
3 months (until 2021.5.24) | 9 months (until 2021.12.24)

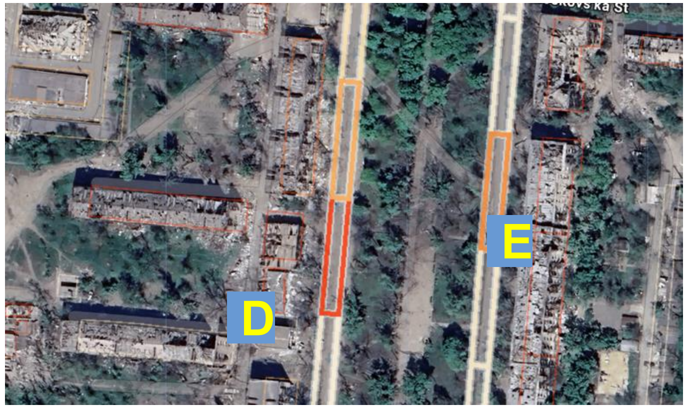
Detection:

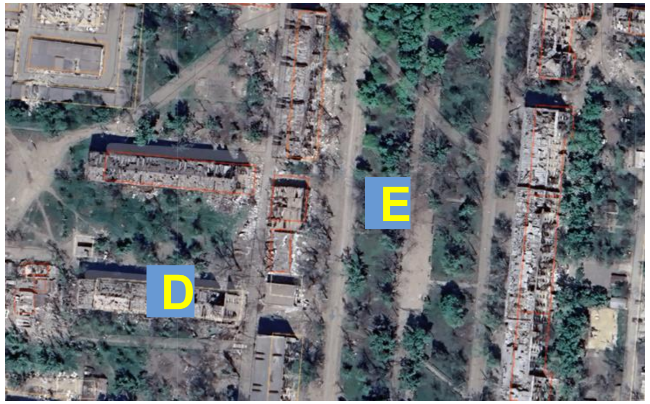
The Inland city of Kharkiv experienced significant seasonal disturbances, showing a gradual and uniform increase over time, peaking in summer. After removing the vegetation layer, the results improved, suggesting a potential impact on t-test outcomes, especially for roads.

T-test Images with Detections

Mariupol 2021.2.24 – 2022.2.24 vs. 2022.2.24 – 2022.3.24







- In Area A, building damage within dense vegetation was successfully detected.
- In Area B, major damages were captured, with minor overflow.
- In Area C, gaps between row houses were clearly distinguished.
- On roads, in Area D, structural road damage was accurately identified.
- However, in Area E, a functioning road was misclassified as moderately damaged.

T-test Validations with UNOSAT Gound Truth Dataset

```python
# label ground truth & pred_damaged
ukr7_all = ukr7_all.copy()
ukr7_all['gt_damaged'] = ukr7_all.geometry.apply(
    lambda geom: ukr7_unosat.intersects(geom).any()
)

ukr7_all['pred_damaged'] = ukr7_all.geometry.apply(
    lambda geom: ukr7_gee.intersects(geom).any()
)
```

label each object with ground truth and predicted damage status based on spatial intersections with the UNOSAT and T-test outcome datasets.

// **Mariupol with 99%**
confidence interval
TP (True Positive): 3362
FP (False Positive): 5800
FN (False Negative): 1831
TN (True Negative): 8695

Precision: 0.3670
Recall: 0.6474
Specificity: 0.5999
Accuracy: 0.6124
F1 Score: 0.4684

a great balance between recall and precision, with a notable improvement in overall performance.

// Mariupol with 95%
confidence interval
TP (True Positive): 3908
FP (False Positive): 7932
FN (False): 1285
TN (True): 6583

Precision: 0.3301
Recall: 0.7526
Specificity: 0.4528
Accuracy: 0.5318
F1 Score: 0.4589

// Kharkiv with 99%
confidence interval
TP (True Positive): 689
FP (False Positive): 57286
FN (False): 182
TN (True): 58532

Precision: 0.0119
Recall: 0.7910
Specificity: 0.5054
Accuracy: 0.5075
F1 Score: 0.0234

successfully identifies most of the truly damaged buildings (high recall), but the large number of false positives (very low precision) mostly likely due to less 1% damage in reality , results in poor overall performance.

// Kharkiv with 95%
confidence interval
TP (True Positive): 751
FP (False Positive): 72650
FN (False): 120
TN (True): 43168

Precision: 0.0102
Recall: 0.8622
Specificity: 0.3727
Accuracy: 0.3764
F1 Score: 0.0202

Limitations & Future Development

- Significant natural disturbances and possible satellite image stitching gaps in cities with different locations
- Suggesting **additional satellite image preprocessing** to reduce natural noise and improve robustness

- A "spill-over effect" was observed: false positives were typically located within one to two Sentinel-1 pixels of truly damaged structures.
- Errors may stem from satellite resolution limits leading to algorithmic misclassification, unclear definitions of war-and warfare related damage to buildings and roads, or mismatches with the UNOSAT ground truth, and the coarse 10-metre resolution, where buildings occupy only one or two pixels, inflating false positive rates.
- Suggesting **introduction of high-resolution imagery,** precisely **defining war-related building and road damage** to improve classification consistency.

- Unlike natural disasters, which typically occur as single events, warfare damage accumulates over months or even years.
- Temporal mismatches between our post-conflict imagery and UNOSAT's single-date assessments may have contributed to higher false positive rates, particularly in Kharkiv.
- Suggesting **shortening the analysis window, incorporating multi-temporal change trends, and adjusting the t-test threshold.**

- The 43.4% missingness of subtype data in building footprint limited accurately classifing damaged building types.

- The missingness of road footprint limited giving quantitative validations to damage road precision model performance.

- Effectively support large-scale preliminary screening
- Used by international organizations in post-conflict urban environments, especially in Mariupol.

↓

- Introduction of high-resolution imagery
- Refined algorithms and analysis timeframe window
- Accuracy by damage definitions
- Considering the incorporation of supervised learning techniques.

Thanks for Listening!

CASA 0025 - Group: SixQL

Data Preprocessing: Tianqi Chu, Ruiya Lin

Data Analysis: Ruiya Lin, Yunqian Yao, Siyuan Feng

User Interface Design: Yujia Ma, Yiyao Cui

Documentation and Report Integration: All