# A Framework for Semi-automatic Collection of Temporal Satellite Imagery for Analysis of Dynamic Regions

Nicholas Kashani Motlagh Aswathnarayan Radhakrishnan Jim Davis
Department of Computer Science and Engineering
Ohio State University
{kashanimotlagh.1, radhakrishnan.39, davis.1719}@osu.edu

Roman Ilin
AFRL/RYAP
Wright-Patterson AFB
roman.ilin.1@us.af.mil

## Abstract

*Analyzing natural and anthropogenic activities using remote sensing data has become a problem of increasing interest. However, this generally involves tediously labeling extensive imagery, perhaps on a global scale. The lack of a streamlined method to collect and label imagery over time makes it challenging to tackle these problems using popular, supervised deep learning approaches. We address this need by presenting a framework to semi-automatically collect and label dynamic regions in satellite imagery using crowd-sourced OpenStreetMap data and available satellite imagery resources. The generated labels can be quickly verified to ease the burden of full manual labeling. We leverage this framework for the ability to gather image sequences of areas that have label reclassification over time. One possible application of our framework is demonstrated to collect and classify construction vs. non-construction sites. Overall, the proposed framework can be adapted for similar change detection or classification tasks in various remote sensing applications.*

## 1. Introduction

The exponential increase in remote sensing data availability has opened up another domain to apply deep learning techniques. However, data-hungry supervised learning techniques for remote sensing tasks require large amounts of labeled data. This need is further exaggerated when classes of interest span larger regions or longer periods of time.

Concerning the raw image data needed, satellite imagery providers, such as Planet Labs [1], capture worldwide imagery each day. This deluge of temporal imagery can provide rich insights into many problems, but the imagery lacks contextual information that describes the scenes captured. Researchers must first identify regions that contain classes of interest. After regions are identified, imagery is acquired, and then manual labeling techniques are employed. Unfortunately, manually annotating large amounts of data is a costly and time-consuming process that limits the scale at which labeled imagery can be acquired. This bottleneck has restricted the remote sensing community from leveraging the full potential of deep learning algorithms in applications such as urban planning, agricultural expansion and abandonment, and landscape monitoring.

Volunteered Geographic Information (VGI), such as open crowd-sourced mapping metadata, can be utilized to assist in collecting and annotating datasets. This metadata enables researchers to semi-automatically identify objects of interest in desired spatial ranges to determine where to collect imagery. Unfortunately, the crowd-sourced nature of the metadata creates noisy or incomplete labels in sparsely labeled areas around the world, hindering its use for fully-automated annotation. However, this metadata could still be used as a starting step in the data collection phase to create candidate annotations that can be further modified or filtered by domain experts since it is much easier to edit existing annotations than to create annotations from scratch using raw imagery. Such metadata is used to *semi-automatically* collect labels since manual verification can be employed. OpenStreetMap (OSM) [2] is one such free and open crowd-sourced spatio-temporal database containing geographical and contextual metadata for physical locations around the globe. The exponential growth of the community of OSM contributors and validators in recent years has improved the reliability of OSM metadata, accelerating the use of such metadata to provide unverified ground truth labels for the abundant unlabeled remote sensing imagery available.

We propose a framework that extends the work of [3] to collect satellite image *sequences* based on label reclassifications for changing targets (*e.g.*, farmland converted to commercial landuse or a new reservoir dug out). Our framework automatically extracts geo-coordinates and contextual information of object classes of interest using historical crowd-sourced OpenStreetMap (OSM) metadata and can collect temporal sequences of multi-modal imagery (*e.g.*, RGB and NIR). The collected metadata and imagery can also be manually verified as needed. Hence, our framework enables users to semi-automatically collect temporal imagery worldwide for multiple applications so long as an OSM label exists for the target of interest.

In this work, we demonstrate our framework's utility with the semi-automatic collection of temporal satellite imagery spanning the periods of construction sites. To show an end-to-end application of our approach, we also provide an exploration of a binary classifier to distinguish image sequences of construction vs. non-construction regions.

The outcome of this work provides a tool to advance the remote sensing community by facilitating the gathering of large and diverse datasets based on various reclassifications and temporal behaviors of targets of interest. Again, the focus of this framework is to provide a means of gathering labeled data for use in multiple remote sensing applications.

## 2. Related Work

The evident hindrance in applying deep learning to remote sensing data is the lack of a large number of labeled benchmark traditional computer vision datasets such as ImageNet [4] and COCO [5]. These traditional vision datasets generally include ground-level imagery that gives a first-person view of classes of interest making them ineffective for remote sensing data problems that require an aerial perspective of these classes of interest. Remote sensing problems such as change detection also require temporal image sequences that record the evolution of a class of interest over time. There are a few existing remote sensing datasets that provide temporal views of objects such as the Functional Map of the World (fMoW) [6] dataset that contains imagery with over 100K unique bounding boxes labeling 63 classes of interest, the xBD [7] building damage assessment dataset that contains pre- and post-damage imagery with over 800K unique building annotations and damage level labels, and the Multi-Temporal Urban Development SpaceNet 7 (MUDS) [8] dataset that provides monthly temporal imagery with over 500K unique building annotations. However, all these large remote sensing datasets mentioned above were collected using expensive and time-consuming manual methods of scraping through existing satellite imagery collections to identify feasible areas of interest and then using crowd-sourced labeling platforms for generating bounding box annotations.

In the remote sensing community, there has been a push to use existing crowd-sourced map data such as OSM to assist in the annotation of the abundant unlabeled remote sensing data available. Previous works such as [9, 10, 11] have studied the use of OSM data for land use classification, which is one of the most popular applications with remote sensing data. Furthermore, the worldwide road center-line vector data in OSM is frequently used to assist in training automated road network extraction methods on aerial and satellite imagery (*e.g.*, [12, 13]). The OSM database also contains labels for building types and their polygonal outlines for building classification and detection tasks [14, 15]. Most of these methods used manual approaches to scrape through OSM database to extract ground truth labels for their particular task. These approaches create a bottleneck as manual extraction is infeasible for task-specific data collection spanning vast regions of interest. They also ignored the wide variety of data recorded by OSM, focusing only on the popular OSM labels such as roads and buildings.

In [3], they proposed a framework for a semi-automated collection of satellite imagery of object classes recorded by OSM. However, their framework ignores *historical* data recorded by OSM and *archived* satellite imagery. In this work, we extend the framework of [3] to collect *temporal* satellite image sequences based on label reclassification of object classes using OSM history metadata. This historical data provides a wealth of temporal information containing local updates to physical locations (*e.g.*, an OSM user adds a new parking garage). We leverage this information to automatically determine both spatial and temporal bounds of OSM object classes of interest. Existing remote sensing datasets such as the fMoW dataset are restricted to a set of labeled classes that are available and datasets such as MUDS dataset only provide spatial characteristics (building bounding box annotations) of buildings without any information about the building types or uses. Our proposed framework works using the expansive set of OSM labels that provide both spatial extents and additional metadata attributes describing the physical features (*e.g.*, building types, landuse) of the classes of interest. The existing datasets also provide only a static set of temporal imagery whereas our proposed framework coupled with daily-revisit image collection sources such as PlanetScope enables collecting custom task-specific datasets with required classes of interest mapped by OSM and temporal revisit frequencies (temporal views can be collected daily except on days when PlanetScope imagery is unavailable). The proposed framework opens a wide variety of applications for change detection and classification in remote sensing domains.

## 3. Proposed Framework

In this work, we present a framework that extends the approach of [3] to automatically find classes that change (*e.g.*, from farmland to building) and then collect and label corresponding temporal satellite imagery. Using this imagery, one can quickly verify labels, if required. Figure 1 provides an overview of the different modules in our framework that will be described in the following sections.

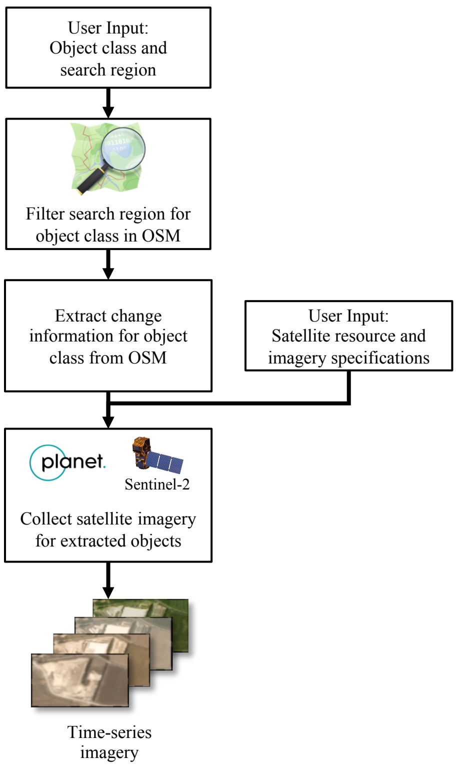


Figure 1: The general pipeline for collecting time-series imagery of selected object classes using OpenStreetMap and satellite resources.

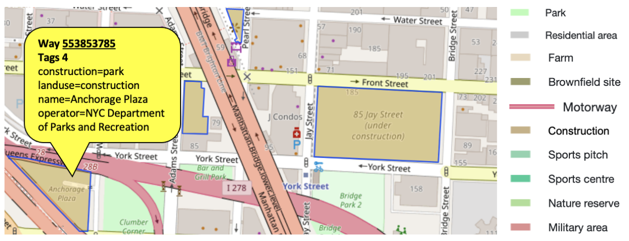


Figure 2: Example area of interest and OSM Map Legend showing various OSM labels. The tags associated with an example OSM construction *Way* are also highlighted in the box.

## 3.1. OpenStreetMap

The crowd-sourced labeling system OpenStreetMap (OSM) has become an increasingly popular metadata source for spatial information. This repository aggregates rich labels and shape information in geo-coordinates recorded by users for physical locations worldwide. Figure 2 shows the richness of OSM labels in an example area of interest.

OSM uses three data structures to represent features on the map: *Nodes*, *Ways*, and *Relations*. A *Node* is defined by a latitude-longitude geo-coordinate pair. For example, traffic lights and stop signs are represented as *Nodes*. Nodes can be grouped to form polygons (*e.g.*, a building perimeter) or polylines (*e.g.*, a road) called *Ways*. A *Way* represents an "area of interest" on the map. Lastly, multiple *Nodes* and *Ways* can be combined to form *Relations*. *Relations* often represent large conglomerations of land (*e.g.*, a neighborhood).

| Key | Value |
|---|---|
| building | hospital |
| building | house |
| building | office |
| landuse | farmland |
| landuse | greenfield |
| landuse | industrial |
| natural | scrub |
| natural | wetland |
| natural | wood |
| waterway | dam |
| waterway | dock |
| waterway | canal |
| ... | ... |

Table 1: Example OSM **"key=value"** pairs.

*Nodes*, *Ways*, and *Relations* can be labeled by the OSM community to provide context to describe the locations in predefined **"key=value"** pairs (*e.g.*, **"building=apartment"**). A key describes the general type of a location, and a value is a refined description of the corresponding key. Users also have the option to specify their own **"key=value"** pair. Table 1 displays an example from the hundreds of predefined **"key=value"** pairs in OSM. The full list of predefined **"key=value"** pairs can be found at [16]. Using these key-value pairs, we can determine the class to which a *Way* belongs. This information will enable us to locate, both spatially and temporally, where and when labels exist and identify any label reclassifications. Due to the crowd-sourced nature of OSM, object labels can be noisy (*e.g.*, inaccurate *Way* geo-coordinates or missing labels). Even so, OSM metadata can drastically reduce the manual effort needed to *verify* the collected data. The following section describes our approach to identify dynamic regions in OSM.

## 3.2. Way Extraction

The first stage of our data collection framework uses historical OSM data to identify *Ways* whose tags were relabeled using changes in **"key=value"** pairs (*e.g.*, **"landuse=farmland"** is relabeled to **"building=house"**). The framework can limit the search to only *Ways* that fall within a user-provided search region (*e.g.*, a particular city or region) and time span. The *Way* extraction pipeline aims to build a "geo-dataframe" [17], a tabular data structure for handling geospatial data, of all ways for object classes of interest within the user-specified search region and time window.

Algorithm 1 outlines the steps in the Way extraction stage. This stage takes the following input from the user:

- An OSM class of interest.
- A start and end date-time window within which to search for relabeled ways.
- A polygon of a search region (in geo-coordinates).
- An OSM history file which contains the archived historical data for the search region.

OSM history files contain logs of all recorded areas of interest in the OSM database and any changes made to the OSM database. OSM history files are available for download at [18]. The first step of the Way extraction stage is to filter out the search region from the history file. The result contains only *Ways* that have been labeled as the class of interest (at least once) and fall within the search region. The framework uses this filtered region to record temporal changes to *Ways* by taking "snapshots" of the state of the OSM database. A snapshot records the state of all *Nodes*, *Ways*, and *Relations* (including their tags) in the OSM database at any point in time. Daily snapshots are taken while stepping through the time window to record newly relabeled *Ways* or changes to existing ones. Once the start and end dates are identified, the framework takes "boundary" snapshots of a relabeled *Way* the day *before* relabeling and the day *after*. These "boundary" snapshots are used to find tags that describe the *Way* before and after its reclassification.

The framework can handle edge cases when a *Way* is deleted and then immediately recreated as a *new* OSM entry. This occurs when a user modifies a *Way* with updated information (*e.g.*, new boundary geo-coordinates). In these cases since there was no reclassification, the framework automatically links the new *Way* to the old *Way's* entry and clears the current end date.

Once all *Ways* are located, each *Way* is saved in a geo-dataframe with the following information:

- Timestamps for the first and last day it was labeled.
- A tag that describes the area before relabeling.
- A tag that describes the area after relabeling.
- The geo-coordinates of the shape of the *Way*.

Our framework's modularity allows using this geo-dataframe as input for the image collection stage or perhaps as data in another application.

**Algorithm 1** Way extraction

```
Input: $start$ - the start date to search
       $c$ - OSM class of interest
       $end$ - the end date to search
       $region$ - a search region
       $history$ - OSM history file containing $region$
Output: A table of relabeled Ways
 1: procedure $ExtractWays$
 2:     # Extract the user specified region from OSM
 3:     $filt\_reg \leftarrow filter(history, region, c)$
 4:
 5:     # Create tables for processing and completed ways
 6:     $processing \leftarrow$ new geo-dataframe
 7:
 8:     $completed \leftarrow$ new geo-dataframe
 9:
10:     # Search through the user specified time window
11:     $date \leftarrow start$
12:     while $date \leq end$ do
13:         # Get state of $region$ on $date$
14:         $snapshot \leftarrow getSnapshot(filt\_reg, date)$
15:
16:         # Add new ways in $snapshot$ to $processing$
17:         $addWays(snapshot, processing)$
18:
19:         # Update ways in $processing$ using $snapshot$
20:         $updateWays(snapshot, processing)$
21:
22:         # Transfer potentially reclassified ways
23:         # Update $way.new\_tag$ for reclassified ways
24:         $transfer(processing, completed)$
25:
26:         # Rollback transfer of incomplete ways
27:         for $way$ in $completed$ do
28:
29:             # Check if $way$ was not reclassified
30:             if $way.new\_tag$ == $c$ then
31:                 Delete $way.new\_tag$
32:                 Delete $way.end$
33:
34:                 # Transfer $way$ back to processing
35:                 $rollback(way, completed, processing)$
36:         $date$++
      return $processing, completed$
```

## 3.3. Image Collection

In the image collection stage, the extracted ways are used to download corresponding satellite imagery. In this framework, we have included the popular Planet Labs [1] and Sentinel-2 [19] platforms to access RGB and NIR imagery. Note that other satellite service providers for other image types (*e.g.*, Synthetic Aperture Radar imagery) could also be employed.

Algorithm 2 outlines the steps in the image collection stage. In addition to the geo-dataframe, this stage takes the following input from the user:

- The desired image modality (*e.g.*, RGB or NIR).
- The satellite imagery provider (Planet or Sentinel-2) and an API key (for a user account to access imagery).
- The desired number of temporal images to extract, spread within the given time window (*e.g.*, 3, 4, ..., all).
- A scale factor used to pad each *Way's* bounding-box.

We provide the option to pad/expand each image sequence to obtain additional spatial context. Images are collected, equally distributed through time, between each relabeled *Way's* start and end date (the start and end dates will always be included). The user can also request all available imagery spanning the time of change.

The first step in the image collection stage is to pad each sequence's bounding box of the polygon by the specified scale factor. Then, a request for the satellite imagery provider is created for each sequence. This request includes the distributed dates, bands, and padded bounding-box. Lastly, each request is submitted to the satellite imagery provider's API. Once the provider processes all requests, temporal imagery is downloaded. At this point, a labeled dataset of targets selected over time is available for the desired task.

**Algorithm 2** Image Collection

```
Input: $t$ - table of relabeled Ways
       $s$ - a scale factor to pad the bounding-box
       $p$ - a satellite imagery provider
       $b$ - a list of electromagnetic bands desired
       $n$ - the desired number of images
Output: Temporal imagery
procedure $ImageCollection$
    # Collect imagery for each way in $t$
    for $way$ in $t$ do
        # Get $way$ bounding box with padding
        $padBox \leftarrow scale(way.bbox, s)$

        # Get the dates $p$ has imagery of $padBox$
        $dates \leftarrow queryDates(paddedBox, p)$

        # Determine $n$ equally distributed dates
        $dates \leftarrow distributeDates(dates, n)$

        # Send a request to $p$
        $reqs \leftarrow requestImgs(padBox, dates, p, b)$

    # Download and save the requested images
    for $r$ in $reqs$ do
        $download(r)$
```

## 4. Demonstration and Application

We demonstrate the data collection framework for the task of extracting and collecting temporal imagery of "construction sites". In this demonstration, a "construction site" is an area where a structure was erected, demolished, or modified (*e.g.*, demolishing or extending a building). This task is also of particular interest to the recent IARPA SMART program [20] that aims to research methods that analyze natural or anthropogenic activities (*e.g.*, heavy construction) using temporal remote sensing data. Our demonstration automatically determines and extracts construction sites from OSM, including the geo-coordinates and contextual information of each site. It also collects RGB imagery from Planet Labs' PlanetScope3 satellites, capturing imagery at 3 meters per pixel. We provide an end-to-end application demonstration by using the data to briefly explore a classification approach to distinguish construction from non-construction sites.

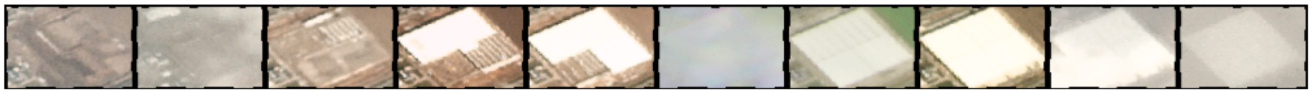
### 4.1. Construction Site Identification and Extraction

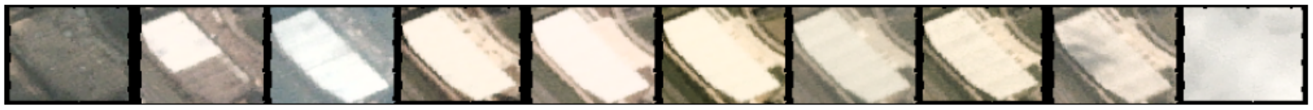
We search for the OSM object classes that represent active construction sites using the standard OSM tags **"building=construction"** or **"landuse=construction"**. We employ the OSM history files for England, France, Germany, and Italy. However, any history files can be employed as long as sufficient crowd-sourced data is available. We selected the time window 2017-02-19 (the day Planet Labs launched *daily* imagery) to 2020-08-22. The framework automatically extracted construction sites across each country, yielding a geo-dataframe of 24,537 construction sites, including their spatial, temporal, and contextual information. An example of three extracted construction sites in the geo-dataframe is shown in Table 2.

Due to the 3-meter resolution of PlanetScope3, the following preprocessing steps were applied to the geo-dataframe before collecting imagery to ensure adequate construction could be visually observed. Sites whose longer image edge was less than 45 pixels (135 meters) or whose shorter image edge was less than 25 pixels (75 meters) were removed. Furthermore, construction sites that did not have an image available within the first 15 days or the last 15 days of construction were discarded. This is because Planet imagery is sometimes unavailable on certain dates due to missing imagery. We collected 30 frames with no padding for each construction site in the preprocessed geo-dataframe

| ID | Start Date | End Date | Previous Tag | Final Tag | Bounding-box Geo-coordinates |
|---|---|---|---|---|---|
| 1 | 2019-02-04 | 2019-10-22 | None | office=company | (12.080231, 47.8803319, 12.0828047, 47.8816893) |
| 2 | 2018-09-19 | 2019-12-23 | None | building=warehouse | (6.830465, 50.6823455, 6.834317, 50.6847926) |
| 3 | 2017-04-13 | 2019-03-11 | landuse=brownfield | landuse=residential | (2.3887916, 48.7850483, 2.3923603, 48.7895063) |

Table 2: Example entries in the geo-dataframe generated by the construction *Way* extraction stage.


(a) Extracted image sequence for ID 1.


(b) Extracted image sequence for ID 2.

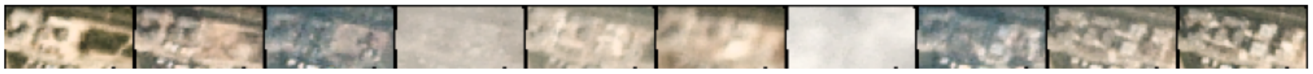

(c) Extracted image sequence for ID 3.

Figure 3: Three image sequences of ten frames corresponding to each identified construction site in Table 2.

using the framework. This process yielded 1,572 construction image sequences. Figure 3 displays 10 frames from image sequences of the 3 extracted construction sites from Table 2. Note that Planet Labs applies black pixels along the periphery where faulty pixels were captured.

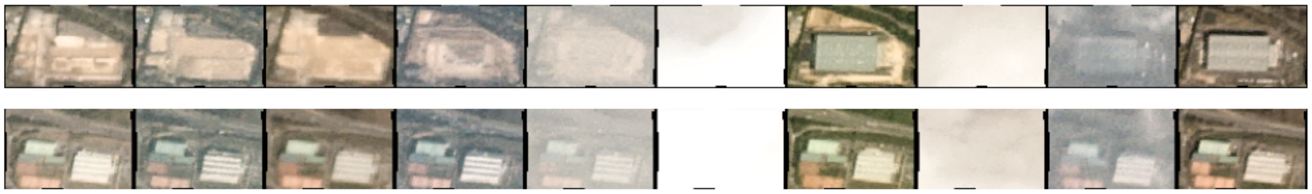
## 4.2. Construction vs. Non-construction Classifier

Given our construction dataset, we provide an example application that classifies construction sites vs. non-construction sites. We analyzed how the number of frames within a sequence affects classification performance.

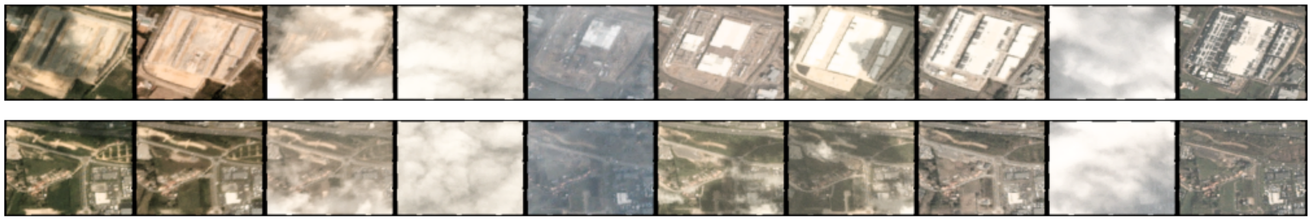
**Dataset.** We used our extracted construction dataset as the positive class and then collected a corresponding non-construction negative dataset. For each construction sequence, a non-construction sequence was collected using the shifted geo-coordinate bounding-boxes of the corresponding construction site. We sequentially shifted each construction sequence's bounding-box right, up, left, or down (random selection) until the shifted box did not intersect with any known construction. This technique enforced that the size distribution of non-construction examples matched the distribution of construction examples. Note that some construction sequences did not have a viable non-construction example nearby. We similarly gathered 30 temporal images from PlanetScope3 for each non-construction site identified using our image collection pipeline. This process yielded 1,520 non-construction image sequences. Each image in a non-construction sequence was captured on the same day as its corresponding construction counterpart. This step ensured that we had equivalent imaging conditions (*e.g.*, similar cloud coverage). We show in Fig. 4 three different construction sequences and their corresponding non-construction counterparts.

Prior to training, we resized the longest side of each image to 50 pixels for all positive and negative examples while maintaining the aspect ratio. We zero-padded (above and below, or left and right) the rest of each image such that it was 50x50 pixels. The train, validation, and test sets were formed by splitting each country's positive and negative examples by $67\%, 16\%, 17\%$, yielding 2,056, 511, and 525 total examples, respectively. Therefore, each set contained a proportional representation of each country's positive and negative examples to combat geographical bias.

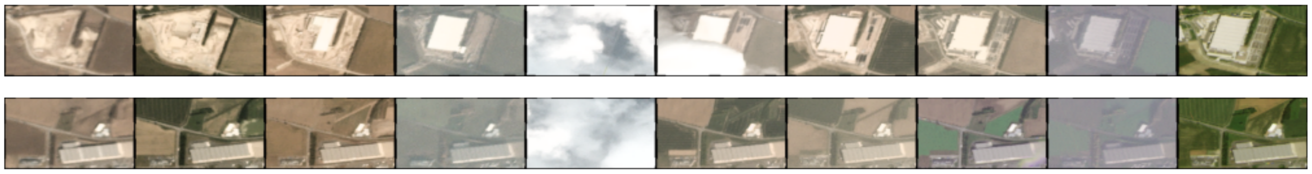
**Classifier.** We analyzed variants of the flexible SlowFast network [21] to study the effects of temporal information on our classification task and provide a baseline. SlowFast networks are a general class of video classification networks that feed sampled clips of a video through two channels: a slow and fast path. The intuition behind the two-path architecture is that the slow path captures long-term spatial features while the fast path focuses more on rapid temporal features. A single Slow network is identical to the SlowFast network but without the fast path. This style of the network provides an ideal framework to analyze different



(a)



(b)



(c)

Figure 4: Three examples of a raw construction sequence (top) and its corresponding non-construction sequence (bottom) in chronological order (left to right).

temporal samplings of the data. Training employed random horizontal flipping for data augmentation and selected the checkpoint with the highest accuracy on the validation set.

We first trained a Slow network for sequences of length 2, 3, 5, 10, and 30 frames evenly distributed across time. The results for these experiments are provided in Table 3. When using only two frames, the start and end frames (commonly used for change detection), the network had only 72.2% accuracy. As we continued to increase the number of frames, the scores increased except in the slight decrease case with 5 frames. The use of the full 30 frames yielded the highest performance in all metrics. These results signify that this change detection task indeed benefits from additional temporal information.

| Frames | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| 2 | 0.722 | 0.747 | 0.685 | 0.715 |
| 3 | 0.770 | 0.766 | 0.787 | 0.777 |
| 5 | 0.764 | 0.790 | 0.730 | 0.759 |
| 10 | 0.787 | 0.784 | 0.802 | 0.793 |
| 30 | **0.808** | **0.812** | **0.809** | **0.811** |

Table 3: Slow network results.

| Slow Frames | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| 2 | 0.794 | 0.838 | 0.738 | 0.785 |
| 3 | 0.806 | 0.854 | 0.746 | 0.796 |
| 5 | 0.810 | 0.855 | 0.753 | 0.801 |
| 10 | **0.813** | **0.863** | 0.753 | 0.804 |
| 30 | 0.811 | 0.833 | **0.787** | **0.809** |

Table 4: SlowFast network results.

We next studied the extended SlowFast model's performance using the full 30 frames in the fast path and a variable number of images in the slow path. The results are provided in Table 4. Increasing the number of frames in the slow pathway marginally increases accuracy and F1. The SlowFast model trained with 10 slow frames had the highest accuracy of 81.3%. We can see that all of these SlowFast models have higher precision than the Slow model trained with 30 frames. However, the recall for all SlowFast models was less than the recall of the Slow model trained on 30 frames (80.9%).

The Slow model trained on 30 frames outperformed all SlowFast models in F1, which indicates that a dual-channel network may not be needed for this task. We suspect that the SlowFast model picked up on seasonal changes in image sequences and wrongly correlated them to construction. For

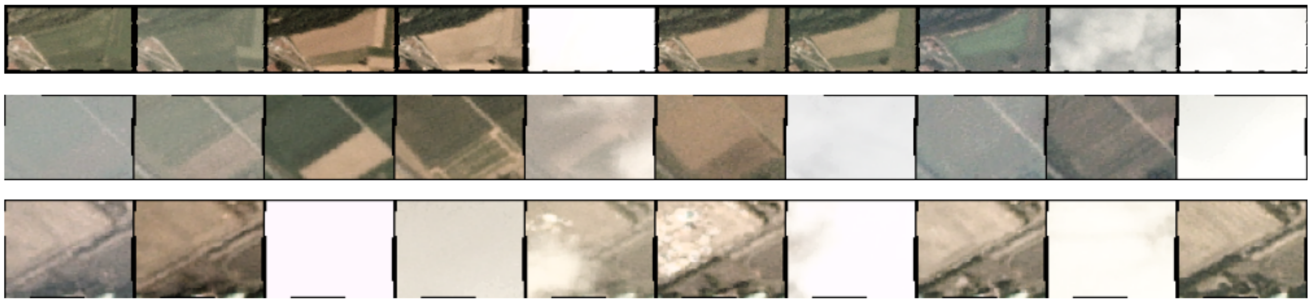


Figure 5: Non-construction sequences incorrectly classified as construction (false positives).

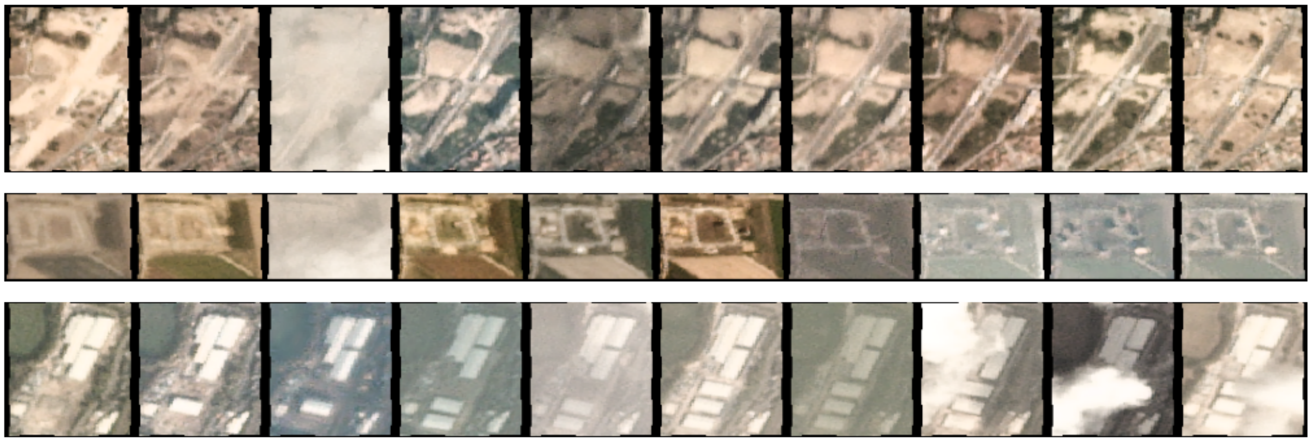


Figure 6: Construction sequences incorrectly classified as non-construction (false negatives).

example, some of the false positives shown in Fig. 5 highlight the effect of seasonal changes on the landscape. We see that the vegetation color turns from green to brown and back to green as time progresses. On the other hand, we also found that the construction of smaller structures was more challenging to identify. In the false negatives shown in Fig. 6, a few smaller construction areas were actually present in a broader tagged region but were missed by the classifier.

As reported in [22], 62% of the 644 peer-reviewed research papers on urban land change algorithms were found to use three or fewer images to measure change. However, our analysis in the construction domain showed that more frames improve classification performance. We again note this classifier is not the primary focus of this work but rather an example application for our framework. Other classification models can also be applied with our framework.

## 5. Conclusion

We presented a general framework to semi-automatically collect temporal satellite image sequences based on the identification of label reclassifications using crowd-sourced annotations and available satellite imagery providers. This data collection framework automatically determines label reclassifications by leveraging OSM history data to collect dynamic regions and corresponding temporal satellite imagery. This framework enables researchers to collect larger, task-specific temporal datasets for remote sensing tasks more efficiently than employing manual crowdsourcing approaches. We demonstrated the capability of the data collection framework for gathering labeled imagery of construction sites. We further showed the applicability of the data collected by our framework by training a classifier to distinguish construction vs. non-construction sites and examining the influence of the number of temporal frames employed. It is expected that this collection tool will enable many more applications and machine learning techniques to be employed with remote sensing data. The construction dataset and python code for the data collection framework can be licensed for use by contacting davis.1719@osu.edu.

## 6. Acknowledgements

This research was supported by the US Air Force Research Laboratory under contract #GRT00054740.

## References

[1] Planet Team, "Planet Application Program Interface: In Space for Life on Earth." https://api.planet.com, 2020. 1, 4

[2] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org." https://www.openstreetmap.org, 2020. 1

[3] A. Radhakrishnan, J. Cunningham, J. Davis, and R. Ilin, "A Framework for Collecting and Classifying Objects in Satellite Imagery," in *International Symposium on Visual Computing*, pp. 295–306, Springer, 2019. 1, 2

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-scale Hierarchical Image Database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, Ieee, 2009. 2

[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *European Conference on Computer Vision*, pp. 740–755, Springer, 2014. 2

[6] G. Christie, N. Fendley, J. Wilson, and R. Mukherjee, "Functional map of the world," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6172–6180, 2018. 2

[7] R. Gupta, B. Goodman, N. Patel, R. Hosfelt, S. Sajeev, E. Heim, J. Doshi, K. Lucas, H. Choset, and M. Gaston, "Creating xbd: A dataset for assessing building damage from satellite imagery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 10–17, 2019. 2

[8] A. Van Etten, D. Hogan, J. Martinez-Manso, J. Shermeyer, N. Weir, and R. Lewis, "The multi-temporal urban development spacenet dataset," *arXiv preprint arXiv:2102.04420*, 2021. 2

[9] B. A. Johnson and K. Iizuka, "Integrating OpenStreetMap Crowdsourced Data and Landsat Time-series Imagery for Rapid Land Use/Land Cover (LULC) Mapping: Case Study of the Laguna de Bay Area of the Philippines," *Applied Geography*, vol. 67, 2016. 2

[10] M. Schultz, J. Voss, M. Auer, S. Carter, and A. Zipf, "Open Land Cover from OpenStreetMap and Remote Sensing," *International Journal of Applied Earth Observation and Geoinformation*, vol. 63, 2017. 2

[11] N. Audebert, B. Le Saux, and S. Lefèvre, "Joint Learning from Earth Observation and OpenStreetMap Data to Get Faster Better Semantic Maps," in *Proceedings CVPR Workshop: Large Scale Computer Vision for Remote Sensing Imagery*, 2017. 2

[12] A. Van Etten, "City-scale Road Extraction from Satellite Imagery V2: Road Speeds and Travel Times," in *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1775–1784, IEEE, 2020. 2

[13] S. Wu, C. Du, H. Chen, Y. Xu, N. Guo, and N. Jing, "Road Extraction from Very High Resolution Images Using Weakly Labeled OpenStreetMap Centerline," *ISPRS International Journal of Geo-Information*, vol. 8, no. 11, p. 478, 2019. 2

[14] P. Kaiser, J. D. Wegner, A. Lucchi, M. Jaggi, T. Hofmann, and K. Schindler, "Learning Aerial Image Segmentation from Online Maps," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 55, no. 11, 2017. 2

[15] W. Zhao, Y. Bo, J. Chen, D. Tiede, B. Thomas, and W. J. Emery, "Exploring Semantic Elements for Urban Scene Recognition: Deep Integration of High-resolution Imagery and OpenStreetMap (OSM)," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 151, 2019. 2

[16] OpenStreetMap Wiki, "Map Features." https://wiki.openstreetmap.org/wiki/Map_features, 2020. 3

[17] K. Jordahl, "Geopandas: Python tools for geographic data," *URL: https://github.com/geopandas/geopandas*, 2014. 3

[18] OpenStreetMap Wiki, "History API and Database." https://wiki.openstreetmap.org/wiki/History_API_and_Database, 2020. 4

[19] European Space Agency, "Copernicus Sentinel-2 data [2020]." https://sentinel.esa.int/web/sentinel/missions/sentinel-2. 4

[20] IARPA, "Space-Based Machine Automated Recognition Technique (SMART) Program." https://www.iarpa.gov/index.php/research-programs/smart/smart-baa, 2020. 5

[21] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast Networks for Video Recognition," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6201–6210, 2019. 6

[22] M. Reba and K. Seto, "A Systematic Review and Assessment of Algorithms to Detect, Characterize, and Monitor Urban Land Change," *Remote Sensing of Environment*, vol. 242, 2020. 8