Human-guided Flood Mapping on Satellite Images

Jiongqian Liang
Department of Computer Science and Engineering
The Ohio State University
liangji@cse.ohio-state.edu

Peter Jacobs
Data Analytics
The Ohio State University
jacobs.269@osu.edu

Srinivasan Parthasarathy
Department of Computer Science and Engineering
The Ohio State University
srini@cse.ohio-state.edu

ABSTRACT

Flooding is responsible for substantial loss of life and economy. Flood mapping, the process of distinguishing flooded areas from non-flooded areas during and after a disaster, can be very useful in guiding first response resources in a disaster situation, and in assessing flood risk in future disaster scenarios. This paper involves the use of image segmentation methods and human guidance to provide a mechanism for flood mapping. Previous image segmentation methods do not work well in flood mapping because they are designed to segment objects out of an image, where there are only a few objects, e.g., foreground-background segmentation. However, satellite images of flooded areas often contain hundreds to thousands of large and small water areas that need to be identified. Therefore, we design a semi-supervised learning algorithm specifically to tackle the flood mapping problem. We first divide the satellite image into patches using a graph-based approach depending on the proximity and intensity of pixels. We then classify each of the patches in an interactive and incremental way, where each time the user is asked to label a few patches and we learn a classifier to automatically classify other patches into water area or land area. We run our algorithm on satellite images of Chennai, India during the 2015 Chennai flood period. The results show that our algorithm can robustly and correctly detect water areas compared to baseline methods. We compare the segmentation results of post-flood with pre-flood and conduct an effective flood evolution analysis.

Keywords

Flood mapping; Graph-based approach; Semi-supervised; Image Segmentation

1. INTRODUCTION

KDD 2016 Workshop on Interactive Data Exploration and Analytics (IDEA'16) August 14th, 2016, San Francisco, CA, USA.

According to Hallegatte et al. [11], if worldwide flood probabilities remain constant over the next 35 years, rising sea levels, sinking land areas, and growing urban coastal populations are expected to drive annual global flood losses from 6 billion U.S. Dollars per year to upwards of 60 Billion US Dollars per year by 2050.

It is of paramount importance to identify ways to reduce the probability of flooding in coastal urban areas. A first step in achieving this goal is the reliable identification of regions in coastal cities that are most susceptible to flooding. If these regions can be identified, action can be taken to better protect these areas from flooding, and public policy can prevent development in areas that have a high risk of flood damage. Flood mapping allows for identification of areas of high, medium, and low risk of flooding, which can help prevent serious flooding from happening. Another application of flood mapping is the quick identification of areas that have been severely flooded during or immediately after a storm. This information can be utilized to guide first responders to where they are most needed.

In order to conduct flood mapping, satellite images promise tremendous potential in monitoring flood disasters due to their low cost and consistent and repetitive data acquisition capability over large spatial areas [17, 20]. Compared to the sparse *in situ* physical sensing data (e.g., river gauge data and weather station records), satellite images offer a synoptic view of the landscape and provide a comprehensive geo-spatial perspective on flood events. The problem here is how to correctly identify areas flooded areas given high-resolution satellite images.

This problem can be regarded as an image segmentation task, where one wants to segment flooded areas out of the whole region. While image segmentation has been widely studied in the image processing community [19, 7, 5, 1], these approaches cannot directly be applied in flood mapping. On one hand, they mostly focus on background and foreground segmentation and the total number of segments is relatively small. On the other hand, these approaches are usually not scalable on large datasets and cannot work on high-resolution satellite images. Moreover, the difference between flooded regions and other regions can be so subtle that human guidance is required in order to correctly locate floods. To address these difficulties, we propose a semi-

supervised learning method that can interact with humans and incrementally conduct flood mapping efficiently.

In this paper, we explore novel ideas to integrate network-analysis and human guidance for flood mapping, where we use network clustering approaches to divide images into patches and adopt human guidance to interactively label the patches as water and land areas. The method for flood mapping involves segmentation of satellite images of the given city before and after a flood occurred to identify land and water areas. This is followed by a comparison of these pre and post disaster segmentations to identify flooded vs. non-flooded areas. The experiments on satellite images of Chennai, India during the 2015 floods show that our method can more effectively identify flooding areas compared to state-of-the-art approaches. Our method is also much more efficient, which enables real-time incremental learning and provides instant information to help prioritize post disaster repair and relief activities.

The rest of the paper is organized in the following way. Section 2 reviews related literature. Section 3 presents our methodology for flood mapping. Section 4 describes an extensive experiment conducted to show the efficiency and efficacy of our method. We describe ongoing and future work in Section 5. Finally, we provide a summary in the last section.

2. RELATED WORK

The problem of flood mapping satellite images is related to the field of image segmentation while the incorporation of human guidance is connected to semi-supervised clustering. Therefore, we review some existing work on image segmentation and semi-supervised clustering in this section. We also discuss past work from the flood mapping domain.

Image segmentation is a long-standing problem and a wide range of techniques has been developed to attempt to segment an image [19, 7, 1, 5, 3, 2]. Some classic methods for image segmentation involve thresholding. Thresholding-based techniques for grayscale image segmentation pick a pixel intensity T and force all pixels with intensity above T to be one color, while all pixels with intensity below T become another color [2]. Thresholding produces a binary image, and if the threshold T is selected carefully, this binary image can isolate foreground objects from the image background, which can be an effective mechanism for image segmentation [2]. Picking the value of T is the main challenge in thresholding. Many methods have been developed for selecting the threshold pixel intensity T [1, 15, 18, 21, 13]. One common method for picking T, known as Otsu thresholding [1], involves finding the pixel intensity that creates the greatest separation and least overlap between the modes in the pixel intensity histogram; this method works best for images with bi-modal pixel intensity distributions. However, Otsu thresholding is not robust when applied to images with noise because the segmentation produced is merely based on the intensity of each pixel without looking at the pixels nearby. If applied to satellite images, it will generate many tiny spots that do not represent relevant higher level structure in the image.

Other methods of image segmentation include the region merging technique proposed by Baatz et al. [3]. They treat pixels as objects initially, and at each iteration, the two objects are merged that lead to the smallest increase in heterogeneity. More recently, graph-based methods have been introduced for image segmentation. Graph-based techniques formulate the image as a graph, and then use some form of community detection to find a segmentation. Shi et al. [19] create a graph with weighted edges and use the normalized cut criterion to segment the image. Browet et al. [7] also formulate the image as a graph; they use modularity as a criterion to find a segmentation for the image. However, these methods are computationally expensive and are not scalable on large satellite images.

Furthermore, there are some semi-supervised learning approaches for image segmentation [5, 14, 4]. One influential semi-supervised method for image segmentation is the watershed algorithm, developed by Beucher and Meyer [5]. The watershed algorithm allows the user to mark different segments in the image. The algorithm then performs a region growing technique, starting from the user placed marks, that operates on the gradient of the original image. While it allows interaction with users and can conduct image segmentation incrementally, the watershed algorithm requires the user to place at least one marker for each segment, which is inefficient in the scenario of flood mapping on satellite images.

Beyond image segmentation, our problem is also relevant to semi-supervised clustering [10]. Semi-supervised clustering involves the addition of "must-link" and "cannot-link" information into the clustering process. "Must-link" information indicates that two objects "must" be in the same cluster. "Cannot-link" information indicates two objects "cannot" be in the same cluster. Wagstaff et al. [22] show that insertion of "must-link" and "cannot-link" information into the clustering process can lead to improved accuracy and efficiency in clustering. However, our problem on satellite images is quite different from the traditional setting of semi-supervised clustering and we need a more convenient way than "must-link"/"cannot-link" for human to provide supervision.

Flood mapping itself has been the subject of previous work. Wang et al. [23] use Thematic Mapping, a type of earth observing sensor, to identify land and water areas before and after flooding, followed by the use of a classification algorithm to identify flooded and non-flooded areas. Henry et al. [12] use Advanced Synthetic Aperture Radar (ASAR) data for flood mapping. These methods both rely on data sources from earth-observing satellites (landsat 7 and Envisat respectively). These data sources are not always available at the time of a disaster. For example, Envisat, the satellite that provided the ASAR data used in the paper by Henry et al., is no longer in operation. Moreover, these methods do not support interactions with ordinary users and cannot update the results incrementally.

3. METHODOLOGY

To effectively solve our problem and overcome limitations in prior work, we state the following desiderata:

- **Fast flood mapping:** Conduct efficient/scalable flood mapping for large satellite images. Efficiency is necessary to facilitate interactive learning and it is also vital if the method is used to help guide emergency first responders in a flood disaster.

- **Guided by human:** Incorporate guidance from humans to achieve better results.

- **Easy to use:** Ordinary users can easily use the method and conveniently provide supervision.
- **High quality results:** Generate effective flood mappings that can be easily interpreted.

Building on these desiderata, we propose a novel method for flood mapping. Our method first preprocesses the satellite images and then detects water areas from satellite images in an interactive fashion using human guidance. Then by comparing the water areas pre and post disaster, it can identify the flood areas. We describe the method in detail below.

3.1 Preprocessing

To label areas of a satellite image as either land or water, we need to decide on a primary unit for labeling. A straightforward way is to treat each pixel as a unit and conduct pixel-based labeling. The drawback is that we lose the information derived from geographic correlations, and the results will tend to be noisy. The labeling results will involve fuzzy and blurring boundaries, and there might be many small spots. Also, it is difficult to label one pixel manually. Another alternative is to conduct uniform grouping, which involves treating the image as a grid with squares of uniform size. However, without using the intensity information of the image, this grouping can go across boundaries (many patches include both water and land), which is not desirable. In this paper, we adopt a graph-based approach for patch generation, which is efficient and can not only effectively detect regions of different sizes, but can also avoid generating regions across land-water boundaries.

3.1.1 Graph Construction

Graph-based segmentation has been widely studied in the literature [19, 9, 6, 7]. In this paper, we convert the image into an undirected graph following the approach proposed by Cour et al. [8]. Each pixel of the image is treated as one node and each pixel has edges to nearby pixels within the distance of d_{max}, where d_{max} is a user-defined parameter. The weight of the edge between pixel i and pixel j is defined using the following approach.

$$w_{ij} = \begin{cases} e^{-\frac{d(i,j)^2}{\sigma_x^2} - \frac{|F(i)-F(j)|^2}{\sigma_i^2}} & \text{if } d(i,j) < d_{max} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

where $d(i,j)$ is the Euclidean distance between pixels i and j and $F(i)$ is a feature vector evaluated at pixel i. The feature vector can be the scalar intensity value or the RGB values of an image. σ_x, σ_i and d_{max} are parameters controlled by the user.

Note that the number of nodes in the constructed graph n is equal to the number of pixels in the image, and the number of edges is $m = k * n$, where k is a small constant factor depending on the setting of d_{max}.

3.1.2 Graph Clustering to Generate Patches

After we construct the graph from the image, we cluster the graph. Since the satellite image usually contains hundreds of millions of pixels, we need a highly scalable graph clustering algorithm. In this paper, we leverage the off-the-shelf tool Multi-level Regularized Markov Clustering (MLR-MCL) [16], which is an efficient multi-level graph clustering software.

Since the goal of graph clustering is to generate basic units for labeling, we tend to produce a large number of clusters. Empirically, we find the method works well when the average size of clusters is a few hundred pixels. Once we have the graph clustering results, pixels in the same cluster are considered to be in the same patch.

There are many advantages to producing patches using this graph-based approach. This approach can avoid the edges/boundaries of a segment being in a patch (e.g. land and water areas being in the same patch). Also, it becomes very easy to control the number of patches using MLR-MCL. Moreover, MLR-MCL has time complexity linear to the number of edges and is very efficient to run in our graph (the number of edges is proportional to the number of nodes).

3.2 Human-guided Labeling

After generating patches, the next step is to ask the user for a couple of labels. The user will place a few markers in the image to label a few patches that they identify as land/water. To utilize this user-provided supervision, a binary classifier is learnt and then applied to the rest of the unlabeled patches.

3.2.1 Learning the Binary Classifier

In this paper, we use k-NN as the classifier since there are only a few features and they are interpretable. In particular, we define the distance function between two patches i and j as follows.

$$D(i,j) = \left\| \bar{F}(i) - \bar{F}(j) \right\|_2 * \log\left(dist(i,j)\right) \quad (2)$$

Eq. 2 contains two factors. The first factor compares the features of the two patches while the second factor calculates the Euclidean distance between the two patches. Specifically, to calculate the first factor, we average the feature vectors of the two patches respectively and compute the L2-norm of their difference. For the second factor, we compute the centroids of both patches and compute the Euclidean distance between the centroids. To decrease the effect of geographical distance, we take the logarithm of the Euclidean distance.

To classify an unlabeled patch, we find the k most similar labeled patches based on the distance function in Eq. 2. The classification of the patch is then decided by a vote conducted using the labels of these k most similar labeled patches.

3.2.2 Interactive Labeling And Incremental Update

Instead of asking the user to label the patches at one time, we create an interactive environment for labeling. The user is asked to label one patch at one time and our method generates classification results based on the labels currently available. The results are presented to the user in real-time and the user decides whether to label more patches or not. If/when the user provides a new label, the method will incrementally update the results. Our algorithm terminates only when the user does not plan to label more patches; at this point, the result is saved. In practice, we find out that the user usually only needs to mark 2 to 6 patches to generate reasonably good results.

Image Date	Size of Image	σ_x^2	σ_i^2	d_{max}	# patches
11/24/2015	800x444	3	16	2	12946
10/19/2015	4500x2500	2	16	2	69674
10/31/2015	4500x2500	2	16	2	69674
11/12/2015	4500x2500	2	16	2	69674
11/24/2015	4500x2500	2	16	2	69674
12/06/2015	4500x2500	2	16	2	69674
12/18/2015	4500x2500	2	16	2	69674

Table 1: Parameter settings used to construct the graph and generate patches

3.3 Flood Mapping

After obtaining segmentations of an urban area before and after a flood, flood mapping can be performed through comparison of the segmentations. We treat the satellite images before the flood as a baseline and compare satellite images during and after the flood with this baseline. Areas that are not segmented as water before the flood, but are segmented as water after the flood are considered flooded areas.

4. EXPERIMENTS AND ANALYSIS

We run our algorithm on real-world satellite images and conduct analysis in this section.

4.1 Dataset and Baseline

We use satellite images of Chennai, India during the 2015 South Indian Floods [1]. In total, we collect six satellite images during the flood, one for every twelve days, shown in Figure 1.

We compare our algorithm with some state-of-the-art algorithms for image segmentation: 1) Watershed algorithm [5]; 2) Normalized cut algorithm [19]; 3) Graph-based image segmentation with post-processing. The method for generating patches in the 3rd baseline is the same method used to generate patches in our method. However, the second step of the 3rd baseline method is purely unsupervised; the step involves continued merging of nearby patches based on the similarity of pairs of patches until the designated number of patches has been generated.

Considering the fact that some baselines (e.g., Normalized cut algorithm and Watershed algorithm) are very computationally expensive and cannot finish on the large satellite images in a reasonable amount of time, we divide our experiment into two parts. In the first part, we downscale the satellite images and run our method and baselines on them for comparison. As an example, we run all the algorithms on the satellite image of Chennai on 11/24/2015, which is re-sized from 4,500×2,500 to 800×444. We then run our method on the full-size satellite images and conduct further performance analysis.

For the experiments, we implement our algorithm using `Python`. We use the OpenCV API for the Watershed algorithm. For the Normalized cut, we obtain the source code from authors [2]. We also implement the graph-based method with post-processing. Basic information about the datasets and parameter settings for our algorithm are displayed in Table 1.

4.2 Comparing Different Methods

[1]https://en.wikipedia.org/wiki/2015_South_Indian_floods
[2]https://www.cis.upenn.edu/~jshi/software/

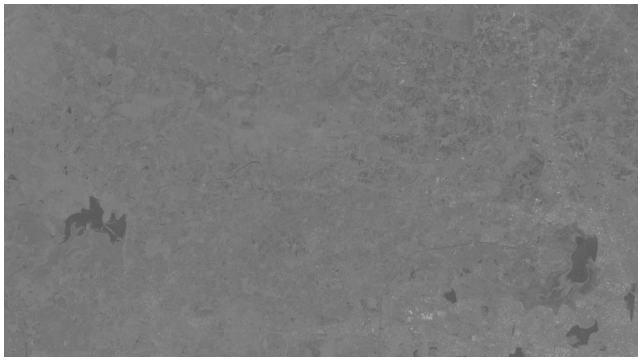
Method	# Markers	Total Time (s)	Interactive Labeling Time (s)
Our Approach	2	30.551	0.057
Watershed Algorithm	11	0.225	0.225
N-cuts Algorithm	0	538.615	0.000
Graph method w. post-process	0	558.220	0.000

Table 2: Running time comparisons of different methods. # markers is the number of markers the human provides for the algorithm.

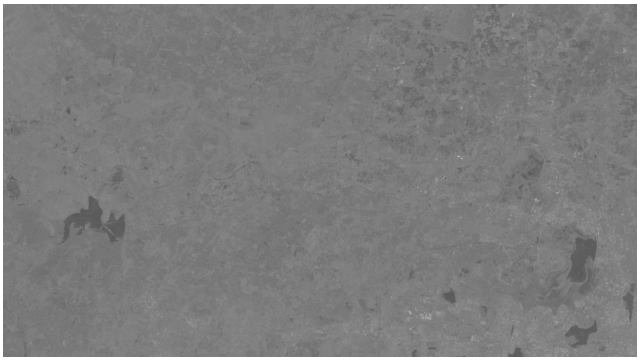
We now compare the performance of our algorithm with the baselines on the downscaled image as shown in Figure 2 (Chennai area on 11/24/2015). The results of segmenting water areas are shown in Figure 3 while the execution time is listed in Table 2. We hereby highlight the following observations:

- Among all the approaches, our method is apparently the best. Our method can clearly identify most of the water areas and even long thin rivers while all other methods fail to do this. Particularly, our method is good at identifying regions of arbitrary shape and it does not limit the size of each segment. We notice that Otsu thresholding [1] might also have similar advantages, but it tends to generate more tiny partitions because its segmentation results only depend on the intensity of each pixel and one pixel can be an individual partition if its pixel intensity is far different from its neighboring pixels [3].
- Compared to the Watershed algorithm, our method produces better results while requiring less effort from humans. The Watershed algorithm seems to correctly capture some boundaries but could not segment out small water areas, including the long thin rivers. In the example shown in Figure 3, we place nine markers in different water areas and two markers in the land areas (see Figure 4). But the segmentation results are still not desirable. On the other hand, using our method, we only need to place one marker in water and one in land respectively (see Figure 4) and the results are much better than the Watershed algorithm. One of the reasons for this difference is that labeling of the Watershed Algorithm grows from the user marked regions in a local fashion and therefore requires much more manual labels for it to work reasonably well.
- The Normalized Cut algorithm tends to generate over-balanced segments and cannot extract segments of long thin shape (shown in Figure 3(c)). Though it performs well in detecting most of the boundaries, it breaks large areas into pieces that should be in one partition. This can be seen from the split of some large lakes. As a whole, the results are much worse than our algorithm.
- Graph-based segmentation with post-processing in general works well in detecting some large areas (see Figure 3(d)). However, similar to the Normalized Cut algorithm, it cannot detect small water regions, especially those long thin rivers.

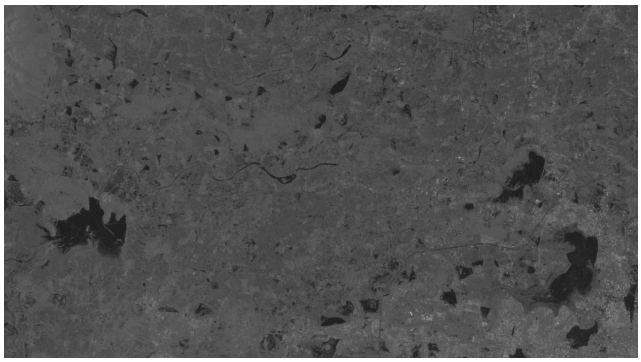
[3]Further investigation shows that Otsu generates two times as many segments as our method on the image.



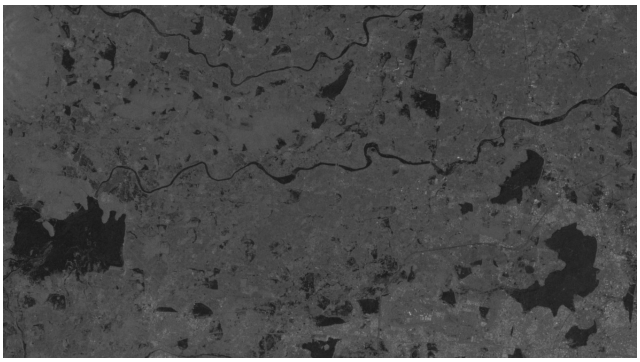
(a) 10/19



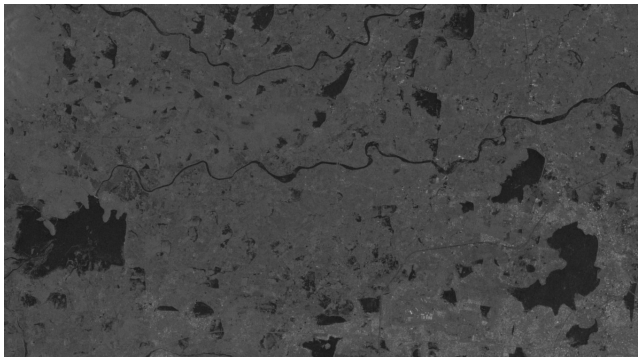
(b) 10/31



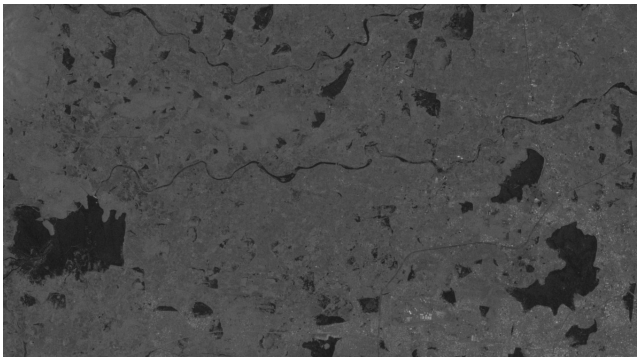
(c) 11/12



(d) 11/24



(e) 12/06



(f) 12/18

Figure 1: Satellite images of Chennai from 10/19/2015 to 12/18/2015. One image for every 12 days.

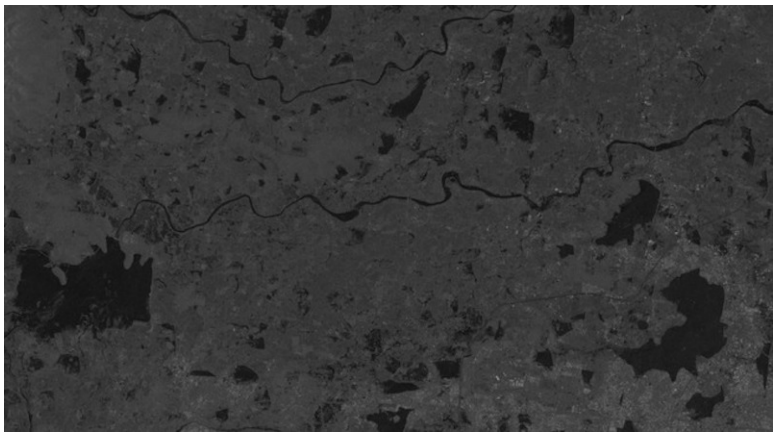
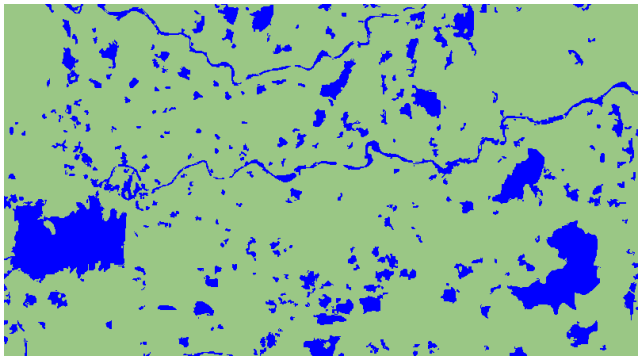
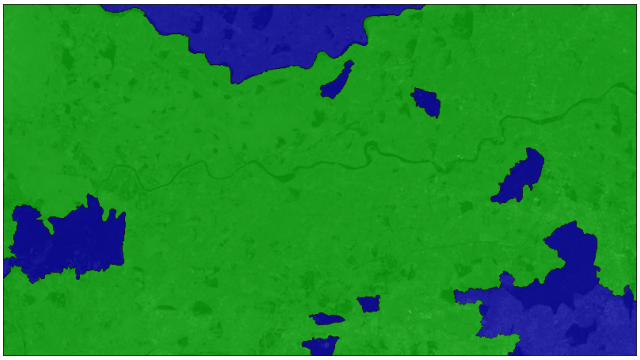


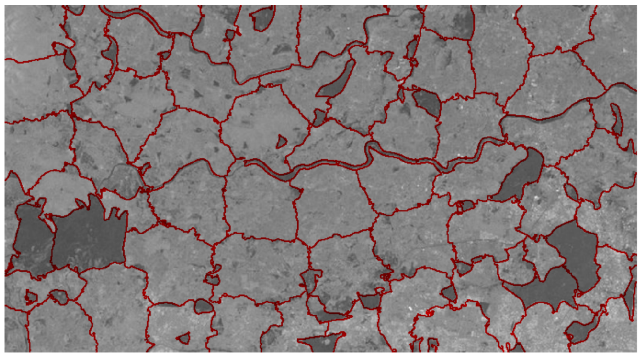
Figure 2: Down-scaled satellite image of Chennai area on 11/24/2015.



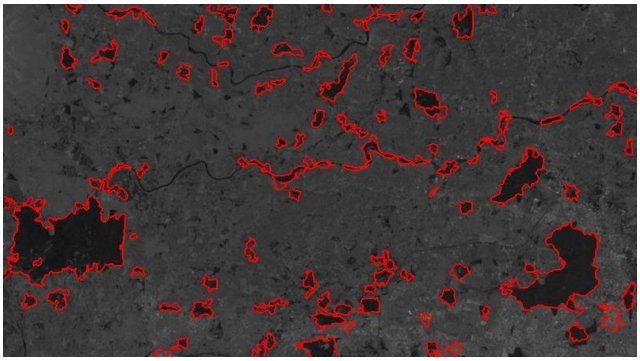
(a) Our Method



(b) Watershed Algorithm

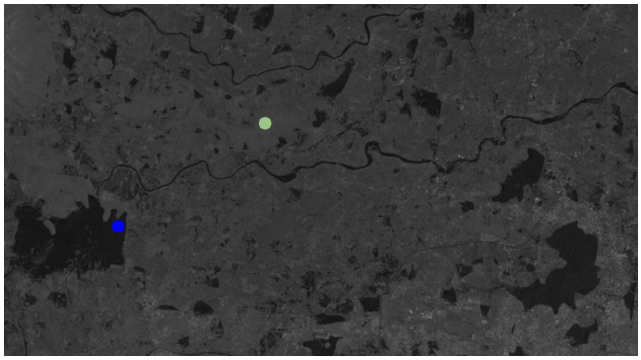


(c) Normalized Cuts Algorithm (100 partitions)

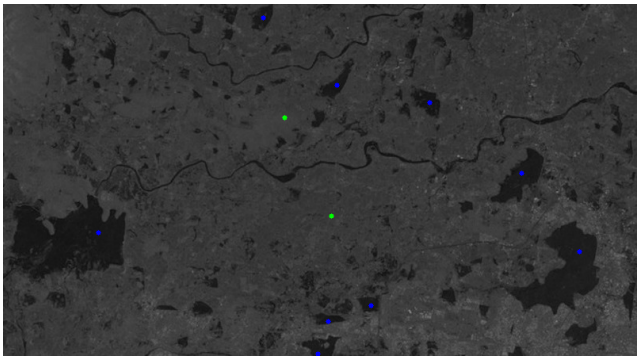


(d) Graph-based Clustering with Post-processing (100 partitions)

Figure 3: Image segmentation results of different approaches on satellite images of 11/24/2015 (down-scaled). (a) is the result of our algorithm, where blue color indicates water areas while green represents land areas. (b) is the result of Watershed algorithm, where blue color indicates water areas and green represents land areas. (c) is the result of Normalized Cut method, where red line marks out the boundaries between land and water areas. (d) is the result of graph-based method with post-processing, where red line also highlights the boundaries between land and water areas.



(a) Two markers provided by the user to our method



(b) Eleven markers provided to Watershed algorithm

Figure 4: Labels that the user provided for the algorithm to learn labeling. Blue points label the areas as water while green points label them as land.

- As shown in Table 2, our method is very fast during interactive labeling and with regard to overall time, it outperforms 2 of the other 3 algorithms. The Normalized Cut algorithm is very slow because it uses spectral clustering and requires computation of the eigenvectors of the Laplacian matrix, which is very computationally expensive. The Graph-based segmentation with post-processing method requires a great deal of computation at the stage of hierarchical merging and is also much slower. Even considering the time of preprocessing for patch generation (30.494 seconds), our method is still much more efficient than the two just mentioned algorithms. Due to the length of patch generation preprocessing, the Watershed algorithm is faster than our method, but our method requires less time during the stage of interactive labeling, which is an important convenience for human users. For example, for one image, we only need to conduct preprocessing once; then, as a result of the efficiency in interactive labeling, the preprocessed image can be interactively labeled by many human users many times easily. This process allows users to find what they consider the 'best' segmentation through trial and error, without long wait times.

4.3 Segmenting out Water Areas on Original Satellite Image

We have shown the advantages of our method compared to other baselines above. Now, we further show the results of our method on all the full-size satellite images in Figure 1. Basic information about the datasets and parameter settings for our algorithm are displayed in Table 1. The segmentation results are shown in Figure 5. From Figure 5, we can observe that our algorithm consistently generates high quality segmentations and is capable of correctly detecting the arbitrary boundaries between land and water. Most long thin rivers, small irregular water bodies, and large wide lakes are correctly extracted.

4.4 Dynamic Analysis for Flood Mapping

While we mainly focus on image segmentation as a method for distinguishing water from land in satellite images above, we now discuss how we adopt the image segmentation method developed to detect flood areas. To this end, we refer to the historical satellite image from before the flood and conduct dynamic analysis.

By simply looking at Figure 5, which shows images of Chennai from 10/19 to 12/18, we can see the water areas greatly increase between 10/31 and 11/12. The water areas start decreasing from 12/06 onward.

To create a flood map, we use the segmentation result from 10/31/2015 as the baseline and compare this segmentation to the segmentations from later dates. Figure 6 presents the dynamic changes of water areas. Red color indicates the areas that change from land into water while yellow color indicates the opposite change. From Figure 6, we can clearly observe that 11/24 and 12/06 have the largest number of water areas; water areas seem to decrease following 12/06. Red areas are likely regions affected by the flood. The flood maps are quite consistent with the fact that the South Indian floods lasted from 11/08/2015 to 12/14/2015.

In addition, we generate two animated gifs and put them on our website [4]. The changes of water areas can be more clearly seen on the animated gifs, revealing the flood surges and recessions.

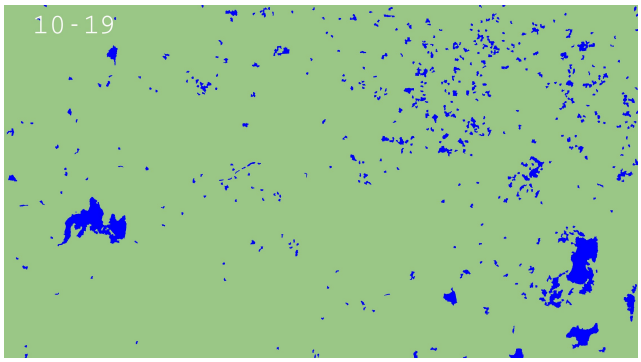
5. ONGOING AND FUTURE WORK

In this section, we discuss some of the directions that we are working on.

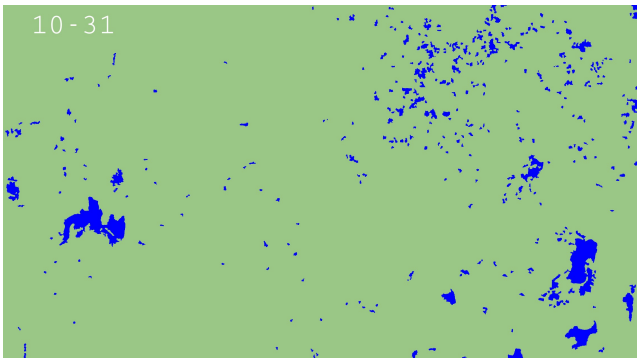
- **Improve the learning models.** While we use very simple k-NN method for classification in this paper, we would like to adopt more advanced classifiers to label patches, such as SVM and neural networks. Meanwhile, more features for each pixel can be leveraged, such as RGB values instead of just intensity. We also want to design an active learning mechanism so that the user will be encouraged to label patches that our algorithm is most uncertain about. This will further reduce the efforts of humans and also improve the flood mapping quality.

- **Crowd Sourcing Experiments.** Some satellite images might be difficult for one person to label and there might be uncertainties and confusions at some parts of images (water or land) due to various reasons, such

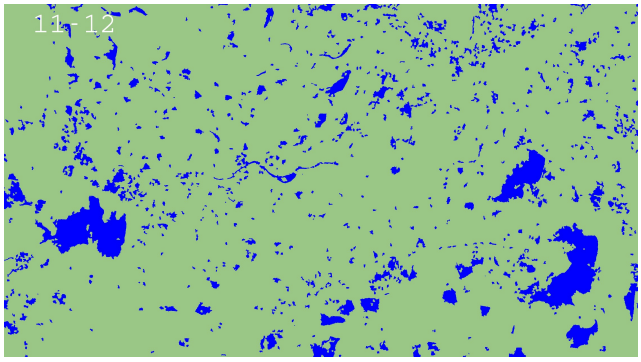
[4]http://web.cse.ohio-state.edu/~liangji/floodmap.html



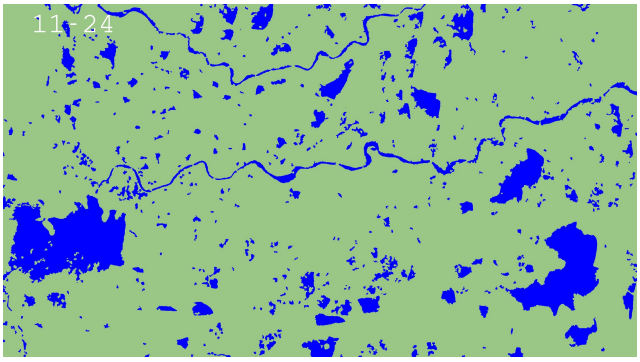
(a) 10/19



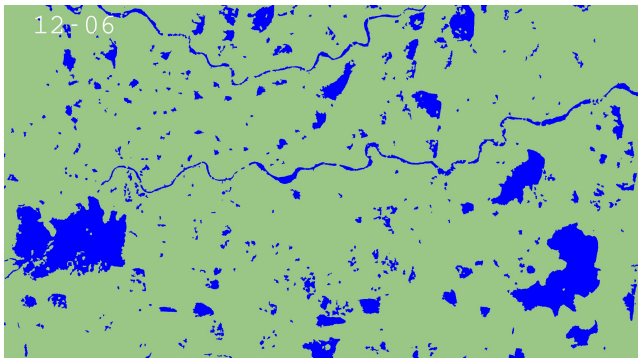
(b) 10/31



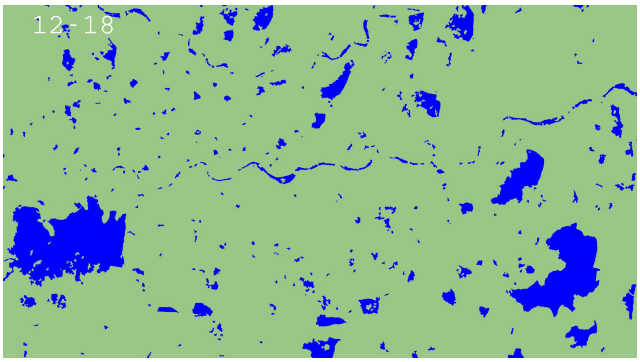
(c) 11/12



(d) 11/24

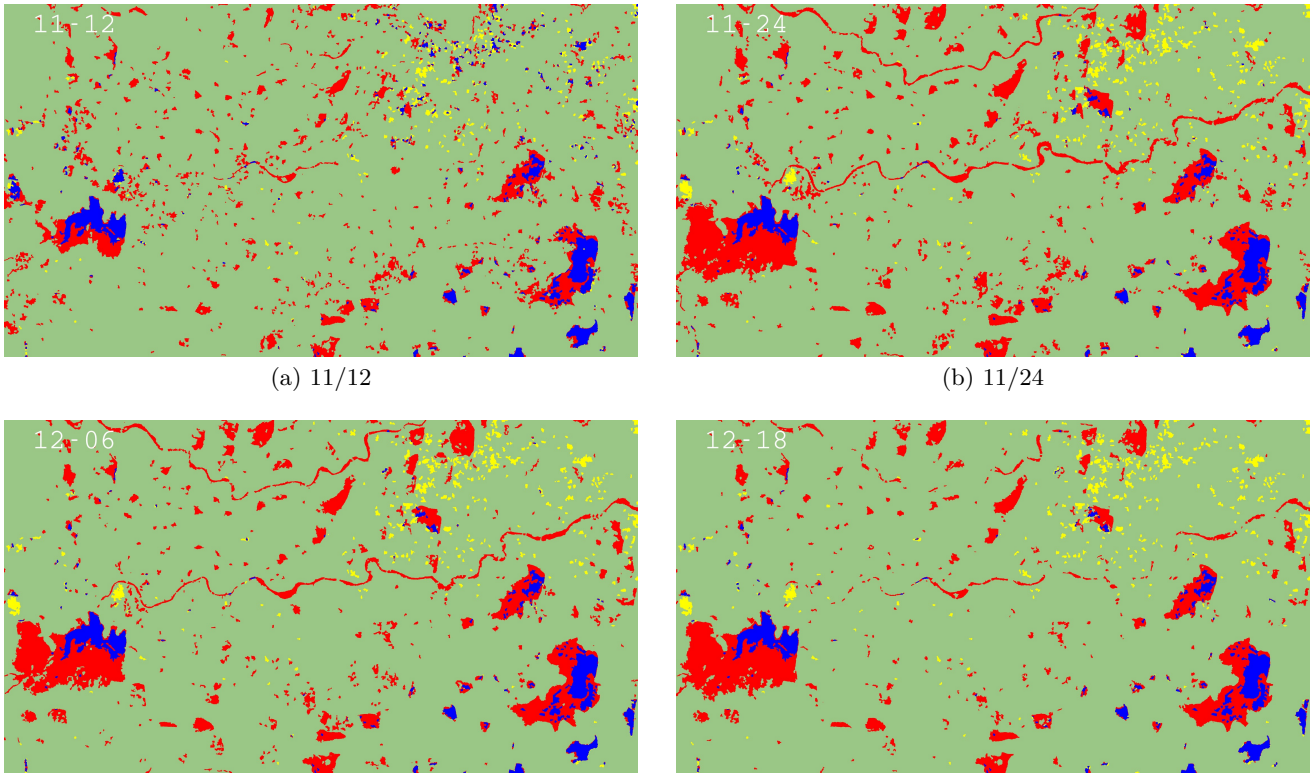


(e) 12/06



(f) 12/18

Figure 5: Results of our segmentation algorithm on satellite images from 10/19/2015 to 12/18/2015. One image for every 12 days.



(a) 11/12 (b) 11/24

(c) 12/06 (d) 12/18

Figure 6: Water area changes from 11/12/2015 to 12/18/2015 using 10/31/2015 as the baseline. One image for every 12 days. Red color indicates areas that were land on 10/30/2015 but were water on the given date, while yellow color indicates areas that were water on 10/30/2015 but were land on the given date. Blue and green represent areas that were originally water or land on 10/30/2015 and remain so on the given date.

as the limitation of resolution. Motivated by this, we plan to lauch a crowd sourcing experiment at platform such as Amazon Mechanical Turk, where we ask different people to help interactively label the satellite images. The segmentation results on the same image are then aggregated. We employ more humans to label those images that involve more conflicts.

- **Incorporating Social Media Information.** During the flood, social media users might publish useful information on social media, which can potentially provide supervision to our method. For example, users might publish tweets on Twitter about the flood in a specific region, and this information can be used as a marker in our method. This means that the information on flood from social media can be used as supervision and labeled markers for the flood mapping approach.

6. CONCLUSION

In this paper, we provide an effective and efficient solution to the flood mapping problem by leveraging human guidance. We generate patches using a graph-based approach and adopt a semi-supervised algorithm involving human guidance to label the patches. Our results show that our algorithm can correctly segment out water and land areas with less noise, compared to other baselines. Further dynamic analysis reveals that it can effectively detect the flooded areas.

Acknowledgements. This work is supported by NSF Award NSF-EAR-1520870 and NSF-DMS-1418265. We also thank Desheng Liu and Jiayong Liang for useful discussions and help with data collection.

7. REFERENCES

[1] A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, Jan 1979.

[2] S. S. Al-Amri, N. V. Kalyankar, et al. Image segmentation by using threshold techniques. *arXiv preprint arXiv:1005.4020*, 2010.

[3] M. Baatz and A. Schäpe. Multiresolution segmentation: an optimization approach for high quality multi-scale image segmentation, 2000.

[4] A. M. Bensaid, L. O. Hall, J. C. Bezdek, and L. P. Clarke. Partially supervised clustering for image segmentation. *Pattern Recognition*, 29(5):859–871, 1996.

[5] S. Beucher and F. Meyer. The morphological approach to segmentation: the watershed transformation. *OPTICAL ENGINEERING-NEW YORK-MARCEL DEKKER INCORPORATED-*, 34:433–433, 1992.

[6] Y. Boykov and G. Funka-Lea. Graph cuts and efficient nd image segmentation. *International journal of computer vision*, 70(2):109–131, 2006.

[7] A. Browet, P. A. Absil, and P. Van Dooren. *Combinatorial Image Analysis: 14th International Workshop, IWCIA 2011, Madrid, Spain, May 23-25, 2011. Proceedings*, chapter Community Detection for Hierarchical Image Segmentation, pages 358–371. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[8] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 1124–1131. IEEE, 2005.

[9] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[10] N. Grira, M. Crucianu, and N. Boujemaa. Unsupervised and semi-supervised clustering: a brief survey. *A review of machine learning techniques for processing multimedia content, Report of the MUSCLE European Network of Excellence (FP6)*, 2004.

[11] S. Hallegatte, C. Green, R. J. Nicholls, and J. Corfee-Morlot. Future flood losses in major coastal cities. *Nature climate change*, 3(9):802–806, 2013.

[12] J.-B. Henry, P. Chastanet, K. Fellah, and Y.-L. Desnos. Envisat multi-polarized asar data for flood mapping. *International Journal of Remote Sensing*, 27(10):1921–1929, 2006.

[13] J. N. Kapur, P. K. Sahoo, and A. K. Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer vision, graphics, and image processing*, 29(3):273–285, 1985.

[14] G. A. Lazarova. Semi-supervised image segmentation. In *Artificial Intelligence: Methodology, Systems, and Applications*, pages 59–68. Springer, 2014.

[15] A. Rosenfeld and P. De La Torre. Histogram concavity analysis as an aid in threshold selection. *Systems, Man and Cybernetics, IEEE Transactions on*, (2):231–235, 1983.

[16] V. Satuluri and S. Parthasarathy. Scalable graph clustering using stochastic flows: applications to community discovery. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 737–746. ACM, 2009.

[17] S. B. Serpico, S. Dellepiane, G. Boni, G. Moser, E. Angiati, and R. Rudari. Information extraction from remote sensing images for flood monitoring and damage evaluation. *Proceedings of the IEEE*, 100(10):2946–2970, 2012.

[18] M. I. Sezan. A peak detection algorithm and its application to histogram-based image data reduction. *Computer vision, graphics, and image processing*, 49(1):36–51, 1990.

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[20] S. P. Simonovic and P. Eng. Role of remote sensing in disaster management. 2002.

[21] D.-M. Tsai. A fast thresholding selection procedure for multimodal and unimodal histograms. *Pattern Recognition Letters*, 16(6):653–666, 1995.

[22] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. *AAAI/IAAI*, 1097, 2000.

[23] Y. Wang, J. Colby, and K. Mulcahy. An efficient method for mapping flood extent in a coastal floodplain using landsat tm and dem data. *International Journal of Remote Sensing*, 23(18):3681–3696, 2002.