A Flash Flood Categorization System using Scene-Text Recognition

Bipendra Basnyat, P.E.
University of Maryland
Baltimore County
b125@umbc.edu

Nirmalya Roy
University of Maryland
Baltimore County
nroy@umbc.edu

Aryya Gangopadhyay
University of Maryland
Baltimore County
gangopad@umbc.edu

Abstract—**Detecting flash floods in real-time and taking rapid actions are of utmost importance to save human lives, loss of infrastructures, and personal properties in a smart city. In this paper, we develop a low-cost low-power cyber-physical System prototype using a Raspberry Pi camera to detect the rising water level. We deployed the system in the real word and collected data in different environmental conditions (early morning in the presence of fog, sunny afternoon, late afternoon with sunsetting). We employ image processing and text recognition techniques to detect the rising water level and articulate several challenges in deploying such a system in the real environment. We envision this prototype design will pave the way for mass deployment of the flash flood detection system with minimal human intervention.**

Keywords: Cyber-Physical System, Smart Environments, Computer Vision, Scene-Text Recognition, Object detection

I. INTRODUCTION

Loss of valuable resources and human lives in flash flood event is one of the nuisances of our daily living. There have been continuous attempts in creating the early alert and warning system to avert or reduce the damage caused by flash-flood event. Until recently, flash flood, and its mitigations have been considered as an engineering problem and primarily left to the civil engineering community for its solution. However, with the upsurge of smart city development and advancement in sensors and (IoT), the paradigm has shifted towards Information communication technology based real-time flash flood detection and early warning systems [11]. This can be attributed to the wide availability of various sensors and the ability to deploy them in rugged environments. Moreover, the technological penetration to interact with the environment via cyber-physical system has expanded tremendously [1], [11]. In other words, advances in sensing and hardware have enabled computers to observe the physical world easily. In addition to sensors, the availability of high definition cameras and streaming devices have provided the vision from remote locations that were otherwise inaccessible. Although the image acquisition has become relatively easy, information extraction and inferring from the images remains a challenging problem. Currently, there is a lack of a reliable and automated image analysis system which has paved the way to a new venue for research in real-time image recognition and object detection. Object detection has been widely used in computer vision especially in image retrieval, and video surveillance [2], [3], [4]. There has been ample research done on face detection, pedestrian-detection, vehicle tracking, etc. [7], [8]. Image recognition has also been exhaustively used in traffic sign detection. One of the reasons behind this is because transportation system relies on road signs which are fairly standard in size, symbols, and colors. Traffic signs are strategically installed along the roadside to convey information to travelers. For example, according to the US Department of Transportation and Federal Highway Administration, the red color is used to depict stop, yield, and prohibition signs. These standard shapes and colors can be used as a-prior knowledge for sign detection and confidence boosting in intelligent transportation domain. Unfortunately, such standardized systems and specific warning signs do not exist in flash flood detection. Rivers and creeks have a flood gauge installed to track flood stage. However, these are rarely automated or connected and also, confined to a specific location. These equipment are manually monitored and often used after the flood event as a record-keeping measure. Furthermore, there is no standard marking or labeling on the scales. These gauges are sometimes as discrete as wall under the bridge with digits or in the units/languages that are local to that region. This makes our contribution and designs unique as we prototype a stand-alone system that can be easily transported and deployed to multiple flash flood prone locations. We also propose a color coding that describes the severity of the flash flood (green: normal, yellow: alert, and red: warning). In this paper, we propose a cost-effective flash flood detection the system that uses the camera (collecting images at different environmental conditions) and water level marker for flash Flood detection. The total cost incurred in parts and materials to build our system is less than one hundred US dollars. We then apply image processing techniques to detect flood levels in a stream and categorize them into various flood levels based on their severity. As a proof of concept deployment, we present one such Cyber-Physical System that is specially designed with a goal to automate the flash flood detection using object detection and digit recognition techniques. We deployed the system inside Patapsco river to detect, monitor and infer the rising water level over a 30 days period (mid-June to mid-July 2017) in different environmental conditions as shown in Figure 1c. The data collected from this prototype deployment has been transferred to an in-house server and used for the image and text recognition, object detection, and inference process.

II. Related Works

We envision that in future, uniformity across different flash flood detection systems and its components will certainly be desirable objectives. However, heterogeneity across system components (e.g., the water level riser position, digit marking, camera angel, environmental conditions, etc.) may exist from one region to another region. To automatically and successfully detect the rising water level with the marker using our proposed camera-based prototype, robust text detection and image processing techniques need to be developed.

Reading text from the natural image is one of the fundamental problems in computer vision with numerous applications. The process of reading relevant text from a natural image is often called "Scene-Text Recognition" [3], and we use this term throughout the paper to describe our work. Scene text recognition can be broken down into two sequential tasks: *text detection* and *text recognition*. However, finding the text from a natural image is very challenging and remains an open problem to be solved. We present some of the previous works that have been done to mitigate this challenge in this section.

In [3], authors proposed a scene text recognition method using part-based tree-structured character detection. They have implemented part-based tree-structure to model each type of character to detect and recognize the characters at the same time. Their final word recognition result was obtained by minimizing the cost function defined on the random field. They have proved the supremacy of their work by presenting more promising results on Experimental results on a range of challenging public datasets (ICDAR 2003, ICDAR 2011, SVT) demonstrate that the proposed method outperforms state-of-the-art methods significantly both for character detection and word recognition. The authors in [4] proposed a real-time system for traffic signs detection, which used template matching based on a new feature expression for geometric shapes, namely, multi-level chain code histogram (MCCH). They have trained their model for different shapes associated with Chinese traffic signs, e.g., circle, triangle, inverted triangle, and octagon. In [5] authors discussed an end-to-end system for text spot localizing and recognizing text in natural scene images and text-based image retrieval. This system is based on a region proposal mechanism for detection and deep convolutional neural networks for recognition. The authors of [14] studied the impact of the map display on the flash flood. They investigated the effectiveness of Mapping software in making public aware of the criticality of flooding zones by more sensitive map display and communicating the environmental threats to stakeholders using an easy-to-use interface. Another relevant work to our research is done by the authors of [15] where they used the "inclined Lidar" technique to detect flash flooding events. Their work is based on the hypothesis that turbidity in water during a flash flood can be used to detect the suspended particles slightly below the surface. In [16], the authors discussed the design and implementation of an IoT based end-to-end emergency and disaster relief system called CROW2. They used Raspberry Pi and Android smart phone to evaluate the performance of their system. The authors of paper [17] presented an RFID outdoor localization system that can localize hundreds of active transponders designed for the tracking of casualties during mass disasters.

One of the significant challenges with the object detection and recognition from an image is to be able to find the area of interest. The problem becomes trickier as we analyze real-world time-sensitive images taken by low-cost low-power sensor/camera. Field camera captures the image in its entirety with most of the content being irrelevant to extract the information. We implemented a hierarchical template matching technique to detect our Region of Interest which enables to perform low-level object recognition on the specific color and digits from the original images in a challenging environment.

Our contributions in this paper are multi-fold. First, we prototype a low-power flash flood detection system (hardware and software) using commercially available off-the-shelf, low-cost sensors. Second, we incorporate the computationally inexpensive scene-text recognition techniques for rising water level detection on real-time. Third, we deploy the system on a riverbed near a flash flood prone area in Ellicott City [13] and collect data over a 30 days period in different weather conditions to evaluate the efficacy of our system and proposed approaches.

III. Problem Domain

Every year, the flood destroys lives and valuable resources throughout the world. The phenomenon was apparent even in the recent flash flood event that occurred in Ellicott City (prototype deployment city) on Saturday, July 30th, 2016 caused by torrential rainfall in the area. The rain yielded a severe flash flood and resulted in two fatalities and millions in property damage [13].

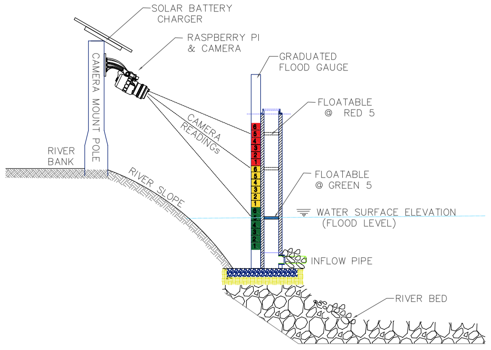
A. Flood

The word "Flood" does not properly resonate with many of us at times. We often see flooding as a small problem within our immediate surrounding such as basement flooding, kitchen overflow, or even a puddle in our neighborhood. Flooding is a lot more than that and one of the major natural calamities[12].

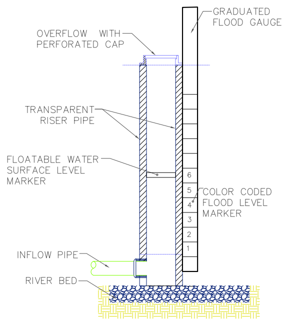
B. Types of Flood

Generally, flood events can be categorized into to three different types. We briefly discuss each of them as follows.

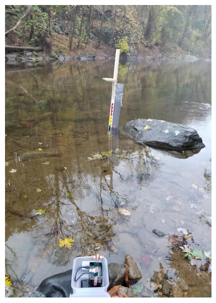
1) **Coastal (Surge Flood):** As the name suggests, this kind of flooding event occurs on in areas that lie on the coast of a sea, ocean, or other large bodies of open water. It is typically the result of extreme tidal conditions caused by severe weather.
2) **Fluvial (River Flood):** Fluvial, or riverine flooding, occurs when excessive rainfall over an extended period causes a river to exceed its capacity. One that occurs suddenly and quickly rises and recedes are often called as **flash flood**. Flash flooding is characterized by an intense, high-velocity torrent of water that occurs in an existing river channel with little to no notice [12].



(a) System Setup


GRADUATED FLOOD GAUGE
OVERFLOW WITH PERFORATED CAP
TRANSPARENT RISER PIPE
FLOATABLE WATER SURFACE LEVEL MARKER
COLOR CODED FLOOD LEVEL MARKER
INFLOW PIPE
RIVER BED

(b) Riser Structure



(c) Deployment (Patapsco River)

Fig. 1: System Components and Deployment

3) **Pluvial (Surface Flood):** A pluvial, or surface water flood, is caused when heavy rainfall creates a flood event in the living areas. Intense rain saturates the urban drainage system allowing for water to flow into streets and nearby structures.

Out of these three types of flood, our current work is about creating a flash flood detection and early warning system for **Fluvial (River Flood)** and more specifically **flash flood** along the river bank.

IV. SYSTEM COMPONENTS

In this section, we discuss the different system components, and sensor hardware and software used to develop our low-cost low-power flash flood detection system. We also draw a schematic diagram of our system before building the prototype as shown in Figure 1a.

A. System Setup

To automate the process of flash flood detection using computer vision and image processing techniques, we needed a time variant image capturing system that can store and track water level in a stream to categorize the flood levels. To achieve this, we developed an end to end system encompassing a prototype deployed in a challenging environment, fetching data and image processing in a back-end server, and sending real-time notifications to a smart phone based on the rising water level. The camera and microcomputer are mounted on a pole which is positioned on a stable part of the river bank. The microcomputer (Raspberry Pi) and camera are powered by the battery which is connected to a solar cell. The camera takes pictures on preset time intervals. For example, in a given setup, the water surface elevation (Flood Level) is at Green 5. Similarly, we have shown two other readings at Yellow 6 and Red 5. The floatable find its level based on the Water level of the river. These readings can be easily converted into actual Water Surface Elevation (WSEL) based on datum conversion at the location of this device.

There are three major components of our system. The first module in the pipeline is design and build of **Image Acquisition unit**, second is the **Image Preprocessing unit**, and finally, there is **Machine Learning unit**.

B. Image Acquisition Unit

Unlike other computer vision research applications, such as traffic sign detection or home address reading, we do not have a readily available flood detection system. To mitigate this, we designed our Flood Gauge/Riser Structure with a color-coded calibrated scale. Engineering details of the Flood Gauge/Riser Structure are depicted in the Figure 1b.

1) **Water Level Riser Structure:** We designed the Water Level Riser Structure from parts found in a local hardware store. This unit consists of a transparent hollow riser, inflow pipe, floatable water marker and an emergency spillway at the top. The bottom of the riser has an inflow pipe; where rising water enters into the system. The second important part of this appurtenance is the floatable water level marker. In this prototype, we used a blue colored floatable plastic that can rise and recede with the water level. Once the water level rises above highest flood markers, water will overflow back to the stream via perforated overflow cap.
2) **Graduated Flood Gauge:** A Graduated Flood Gauge is attached to the Water Level Riser Structure. This is a color-coded Water Level Scale with three color-coded regions and 1-inch demarcation. The scale used in this prototype can measure Water-surface difference up to 18-inches. Figure 1b shows important parts of our Flood Gauge Riser Structure.
3) **Camera and Micro Computer:** The most crucial task for our work is being able to capture time-variant images of rising/receding water level in the stream. As our system is to be deployed in the wild, so we needed a portable image capturing unit. Therefore, we encased raspberry pi micro-computer and a camera into the watertight enclosure with its battery supply. The raspberry pi camera connected to a Raspberry Pi which takes a time-lapse picture with an increment of one minute between images. The rising and receding water enter

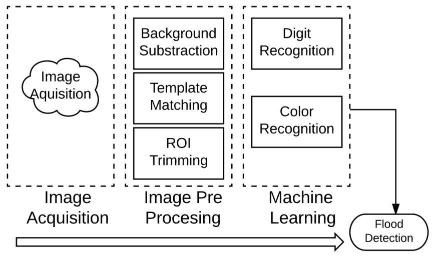


Fig. 2: System Components and Process Flow

and leave the Water Level Riser structure through inflow pipe. Figure 5 upper right corner shows this setup.

C. Image Preprocessing Unit

The main challenge in Scene Text recognition is to be able to find and locate the area of interest with an image. We used various image pre-processing techniques such as background subtractions, template matching and Region of Interest (ROI) trimming to isolate the areas to perform information extraction.

D. Machine Learning Unit

This is the final process in our work to automate the flash flood categorization task. We implemented popular clustering techniques K-Means to separate the most dominant colors. We then used digit recognition techniques to identify the flood level in a stream. Figure 2 summarizes all the components of our flash flood detection system.

E. System Software Architecture

The Micro Computer (Raspberry Pi) runs on Linus Operating system called Debian. We installed MySQL database to store and manage the meta-data such as image time, label, size and deployment location, etc. The data from the Raspberry Pi is extracted using two mechanisms, either by removing the SD card and mounting it to a drive in a computer or via HTTP data pull mechanism. Images and the meta-data are transmitted and stored in the Linux based Centos 6 server with Apache, MySQL and PHP installed on it. We have also provisioned the Raspberry Pi to send the data over WiFi and telenetwork's services such as 3G/4G/LTE etc. The server uses MySQL database to store the labeled data. Images are stored in jpeg file format. Field deployed Raspberry pi transmits the data periodically to the server via HTTP.

V. METHODOLOGY

In this section, we discuss our work and implementation steps in detail. We continuously collected images on various weather conditions and at varying water level. Raspberry Pi camera was set up to take a 2 Mega Pixel image. As a first preprocessing step, the images taken by Raspberry Pi camera were reduced to 100 KB. We then proceeded with the actual image processing technique such as background subtraction, template matching, Region of Interest (ROI) trimming. Detailed explanation these steps are discussed below.

Once the images were pre-processed and ready for recognition task, we implemented two techniques to initiate scene-text recognition and flash flood categorization. The first one is color recognition using basic image processing technique such as histogram and finding most dominant color in the image. The second step is to implement digit recognition which was achieved using the deep neural net.

We started the scene-text recognition by localizing area of interest, i.e., to find the crosshair point between the flood gauge and in our water level marker found by its location within the riser. Once this point was determined and captured by the camera, we performed two levels of template matching (hierarchical template matching) technique as described below.

A. Hierarchical Template Matching

Template matching works by "sliding" the template across the original image. As it slides, it compares or matches the template to the portion of the image directly under it. It does this matching by comparing pixel intensity and the extent to which the template and the portion of the original image match. We used Open CV template matching library to get the matching ROI. The template matching process requires three parameters including the full image where the search is being performed. Based on the relative correlations of these corners, it helps find the four corners. We used the template matching technique to separate areas that are otherwise unnecessary for our information extraction. There are multiple algorithms in the literature to perform template matching. After experimenting these algorithms, we found that the best performing algorithm is TM _CCOEFF_NORMED. The template matching under this algorithm is given by:

$$R(x,y) = \frac{\sum_{x',y'}(T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}} \quad (1)$$

For the main image size of $W \times H$ pixels and template of $w \times h$ pixels, the resulting template boundary drawn is of size $(W - w + 1) \times (H - h + 1)$ pixels. In other words, the new border drawn around the main image is 1 pixel wider in both directions. The function slides through the main image, compares the overlapped patches of size $w \times h$ against the template using the specified method, stores the comparison results, and draws a rectangle around the full image.

We implement the template matching twice as depicted on the Figure 3. Level 1 template matching technique produces input for the second set of template matching task. Since there are two levels of template matching performed one after the other, we refer this as a hierarchical template matching.

Step-1:

For Level 1 template matching, we start with the image ($Image_1$) that has been captured by the Raspberry-Pi camera. This is a full image but with a reduced resolution.

Step-2:

From the full image, we are only interested in the area around our Graduated Flood Marker. Therefore, as a part

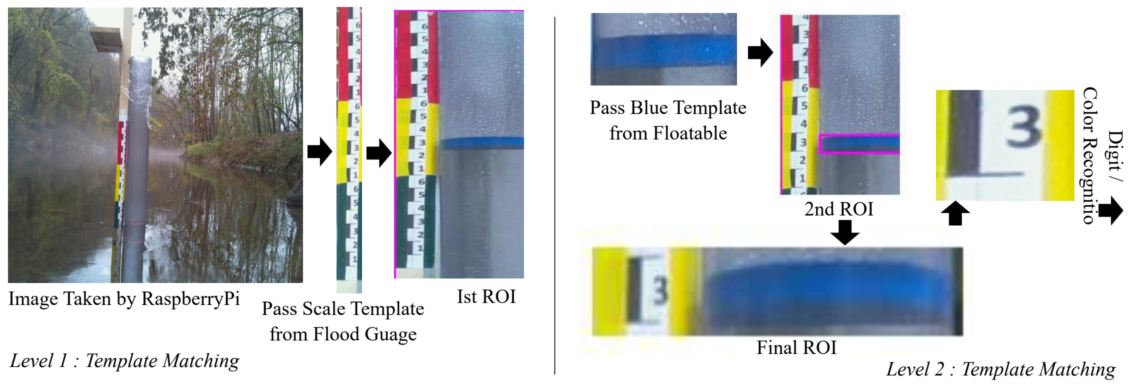


Image Taken by RaspberryPi
Pass Scale Template from Flood Guage
Ist ROI
Level 1 : Template Matching
Pass Blue Template from Floatable
2nd ROI
Digit / Color Recognitio
Final ROI
Level 2 : Template Matching

Fig. 3: Hierarchical Template Matching

of Level 1 template matching, we pass the trimmed image of our flood gauge as an input template. The algorithm involved in hierarchical template matching is given below. Result of the the first hierarchical template matching is depicted in Figure 3., *Level 1 Template Matching*. Algorithmic representation of the steps involved in Level 1 template matching is given below.

Algorithm 1 Hierarchical Template Matching

```
procedure REGION OF INTEREST($ROI_1$)
    $Image_1 \leftarrow$ Camera *Image*
    $ROI_1 \leftarrow$ *Trimmed Scale Image*
    if $ROI_1$ *is Found* then Draw Rectangle around $Image_1$
    $ROI_2 \leftarrow$ *Trimmed Image from* $ROI_1$
    $Image_2 \leftarrow ROI_2$
```

Step-3:

The output from Level 1 template matching was used as an input to the Level 2 template matching. This gave us a closer view of the Water level marker within our Riser Structure. Other pixel level alterations were done to remove the ROI line generated by Level 1 template matching. Furthermore, since template matching finds the matching ROI within the image, the width of ROI was expanded by the width equivalent of the Riser pipe to get a full view of the area for further processing.

Step-4:

Hierarchical template matching was continued on resulting trimmed image from Step 3. But this time the input template is a smaller blue floatable area. The final ROI after the Final Hierarchical Template Matching received. At this point, we have a clear picture of our ROI, and we are ready for the real task, i.e., perform scene-text recognition. Result of the First Hierarchical Template Matching is depicted in Figure 3 , *Level 2 Template Matching*.

B. *Image Segmentation for Color Recognition*

Image segmentation is the foundation of computer vision applications. Its purpose is to partition the image into several independent, meaningful and semantically related regions. An effective and accurate image segmentation algorithm is crucial for many applications, such as content-based image retrieval, object recognition, and object tracking. The objective of our work is to be able to automate the information about Water Level in a stream for early warning and detection system. To that end, we needed to know the color at which Water Level marker is at that time. To achieve this, we used two techniques of color recognition. Both of these methods are described below.

1) Most Dominant Color Using K-Means Clustering: Once the hierarchical template matching was complete the next task for us was to extract most dominant color from the Image. After extracting color values of the trimmed image, the pixels in the image were partitioned and classified into two disjoint clusters (k = 2) via the unsupervised k-means clustering algorithm according to their color. Clustering is done such that the colors of the objects within each cluster are as close as possible to each other. Cluster boundaries are defined by its centroid and the neighboring pixel objects. The centroid for each cluster is the value to which the sum of distances from all the objects in that cluster is minimized. Once the grouping is done, the process is repeated several times to calculate the new centroid of each cluster and subsequently reassign the member objects to the cluster having the closest distance to the new centroid.

We implemented K-Means clustering with given cluster size of two with an expectation that there should be two predominant colors in our final image. As expected, K-Mean cluster identified two predominant colors as shown in Figure 4 below. Figure 4 shows the two distinct color segregation, *blue:* from the template and *yellow:* from the flood gauge.

2) Histogram-Based Color Segmentation: Based on our template and riser structure as well as the Graduated Flood Marker, we know that the color is located on left most corner. Thus we implemented another color segmentation task using

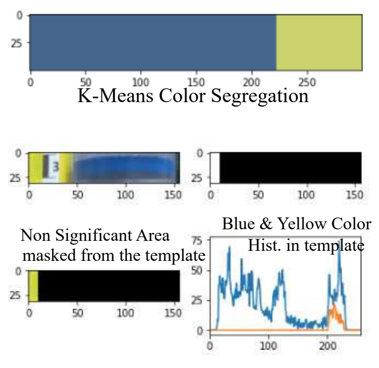


Fig. 4: K-Means Clustering

image masking to "black-out" the colors that are irrelevant to our color extraction technique. This gave us a more focused area of our ROI with unnecessary portions are being masked out. As depicted, in the Figure 4 there are two clear histograms found in the image.

C. *Digit Recognition using Neural Net*

As stated, we wanted to read the flood level to be able to categorize flood severity. Thus, the next task in our pipeline was the machine learning to recognize the number from our final ROI. We implemented neural net using our dataset and the MNIST dataset.

A "neural network" (NN), is a mathematical model or computational model based on biological neural networks, in other words, is an emulation of the biological neural system. It consists of an interconnected group of artificial neurons and processes information using a connectionist approach to computation [9]. A neural network is an interconnected group of nodes, akin to the vast network of neurons in the human brain [10].

To build a digit recognition system, we needed a large dataset with numbers. Unfortunately, such data were not available in public domain, in contrast, we relied on MNIST dataset. The MNIST provides the database of handwritten digits, which contains a training set of 60,000 examples, and a test set of 10,000 examples [7]. Each of these images is 28×28 pixels. Thus they can be seen as 784 dimension vector. This is very popular dataset for experimentation on learning techniques and pattern recognition methods on real-world data.

We altered images captured by Raspberry camera to 28 pixels so that they are in same vector representation as the original MNIST size, i.e., 784D. Furthermore, the MNIST dataset consisted of digits 0 through 9 (10-Class), whereas our Graduated Flood Marker has only digit 1 through 6. Thus we removed digits 0, 7, 8, 9 from the MNIST test and training set. This reduced the original MNIST training set to 36,012 and test set to 6027 images. We then used the feed-forward neural network where the information moves in only one direction, forward, from the input node through the hidden node to the output node.

VI. Evaluation

In this section, we present our results, lessons learned and the challenges faced during the deployment of our device.

A. *Experiments*

Before deploying the system in real flash flood prone environments, we have done controlled experiments both in indoor and outdoor settings simulating a flash flood scenario. To that effect, we started experimenting with our prototype extensively in indoor and outdoor environments with the simulated flash flood as shown in Figure 5. Next, we describe the system setup and lessons learned during this process.

1) **Controlled Flood Simulation.** To validate our proposed scene-text recognition technique and prediction algorithm we simulated rise and recede of water level within the riser structure. We simulated the effect of water level fluctuation in a river by connecting a garden hose to the riser and recording images. This allowed us to iterate and improve the resiliency and capacity of our device, specifically the selection of the floatable material (Water Level Marker) as described below.
2) **Water Level Marker Selection.** While it may seem trivial, Water Level Marker Selection was an important step of our research. Before we settled on our final blue plastic floatable as the Water Level Marker, we tried various options to improve our template matching probability. Following iterations were notable steps forward in the right direction.
    - **First Iteration.** In our first iteration, we thought that the Water Level within the riser would be enough to separate the granularity and give us a matched template location. Contrary to our expectation, as our riser is transparent and there were traces of water droplets around the riser during rain event the template matching algorithm failed.
    - **Second Iteration.** Next, we tried a piece of cardboard as a floatable, but the results were not promising. Since the color contrast was not obvious, the template matching failed on this iteration too.
    - **Final Iteration.** Finally, we decided on a high contrast colored plastic object. This gave us the best result in our image segmentation. Furthermore, since the blue color is not part of the Flood Gauge, we could easily perform color segmentation. All of our setups are depicted in Figure 5. Note in Figure 5, the template matching techniques could not converge until it had a clear pixel separator such as blue floatable.

B. *Color based Flood Level Categorization*

Once the ROI is determined color detection is almost a fail-proof process. The color segmentation process as described is able to identify three different colors that signify the severity of the flash flood, *green: normal*, *yellow: alert*, and *red: warning* level with its partner digit for the actual reading. A simple rule

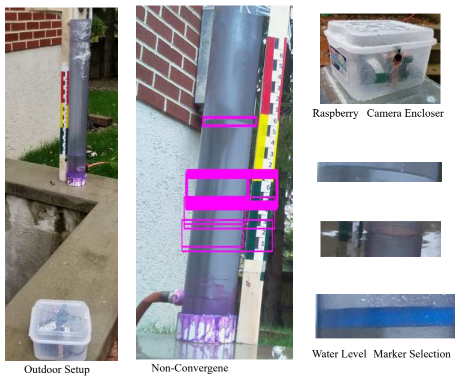


Fig. 5: Outdoor Setup

based algorithm as described below can be used to categorize the flash flood level.

Algorithm 2 Flood Categorization

```
if $Color =$ “Green” then Flood Level ‘Normal‘
else if $Color =$ “Yellow” then Flood Level ‘Alert‘
else $Color =$ “Red” Flood Level ‘Warning‘
```

C. Performance and Resiliency Test

To test the performance and resiliency of the Cyber-Physical System we deployed the system in different weather and site conditions. In this paper we discuss two metrics: **Weather Condition** and **Camera distance** as our parameters in evaluating the performance and resiliency of the system. Figure 6 shows images captured by the Raspberry Pi camera in these conditions.

1) Weather Condition:

The importance of such alert system increases with the increase in weather harshness. To evaluate the resiliency of our system we collected data in varying weather conditions ranging from mild to harsh. We briefly discuss the performance of the device in following weather conditions: such as *Sunny Day, Foggy Day, Drizzling Rain and Heavy downpour*.

- **Sunny Day:** The setup and image taken in a fairly sunny and clear day are shown in Figure 6a. Note in Figure 6a, the template matching techniques could work easily with the floatable. We were able to successfully perform Level 2 hierarchal matching meaning the exact location of flood level was found. Though this is a promising result and can be used for the passive alert system, the outcome of results on such days may not be significant concerning downstream impact and flood alerting system.
- **Foggy Day:** The setup and image taken on an overcast/foggy day is shown in Figure 6b. The success rate of the template matching was comparable to the result from a sunny day and the template matching techniques.
- **Drizzle:** The setup and image taken on a drizzling rainy day are shown in Figure 6c. The template matching technique succeeded in both level 1 and level 2 hierarchy template matching.
- **Heavy Rain:** The setup and image taken on a heavy downpour are shown in Figure 6d. Note in Figure 6d that the template matching technique failed on Level 1 of hierarchal matching as we see multiple boundaries drawn around the ROI. So the setup “As-Is” is unable to depict the flood level or serve its main purpose clearly. Without a clear demarcation of the template area, we cannot any subsequent steps. The result shows that the device needs careful planning and precise location and camera angle for successful template matching.

2) Camera Distance:

Camera location and the distance between the riser and the camera is another critical factor in our deployment. The location of camera and riser structure is dictated by the site condition and expected flood level. If we keep the riser too close to the stream bank, then we may not be capturing the variance in river stage accurately. If we move more inwards into the stream, it may pose a physical danger to both riser and the installer (risk of being swept away) by the gushing water.

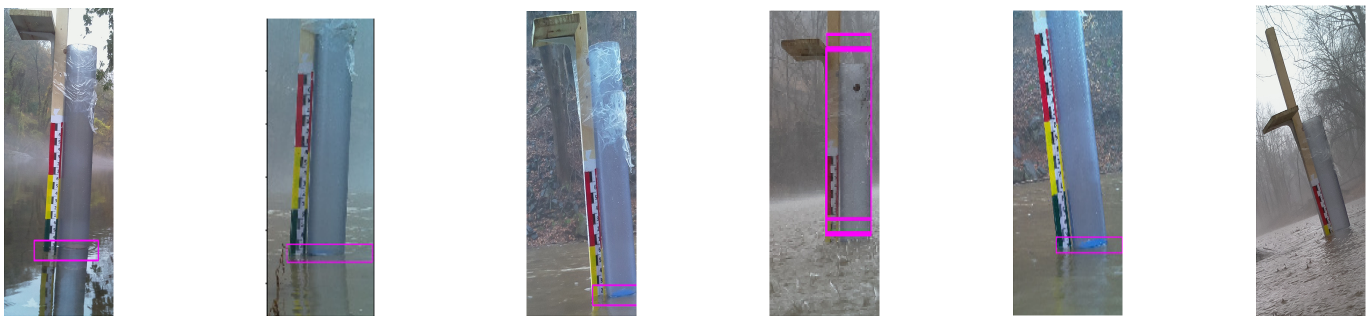
- **Distance-Near:** The setup and image were taken where the distance between the riser and the camera is around 5 ft. Figure 6d shows the setup. It can be observed that template matching performs well-given proximity of the camera and riser structure. However, the desired proximity may be constrained by the landscape and maximum expected flood extents.
- **Distance-Far:** The setup and image taken on where the riser is about 10 feet from the camera, this is shown in Figure 6f. Note in Figure 6f that the template matching technique is unable to draw any ROI on this image. This is the worst condition among discussed here. The floatable and the flood gauge is almost invisible to the camera and hence no template matching.

Based on the observations presented here with, we see that given a proper distance and visibility the system can be improved. Even during a heavy downpour, the template is visible, and it is just a matter of finding optimal riser location and camera angles to the flood gauge.

D. Digit Recognition for Flood Severity

As a proof of concept of the digit recognition model, we used the subset of MNIST dataset. From MNIST dataset we selected images for digits (1, 2, 3, 4, 5, 6) since our flood gauge is marked from 1 through 6. We tested the model with digits from our Graduated Flood Marker. Since there are only six digits repeating over each color, we pass six digits to see the classification performance using MNIST as a training module.

Though, MNIST is a handwritten dataset where as our text is type written (computer printed). We were surprised that our model was able to achieve 33% accuracy in the presence of such heterogeneity in datasets. This is still a promising result for our proof of concept deployment.



(a) Sunny (b) Foggy (c) Drizzle (d) Heavy (e) Near (f) Far

Fig. 6: Performance and Resiliency Test

VII. Discussion

We would like to investigate the following aspects of our proposed system to improve the reliability and fault tolerance.

- The images that are taken at night cannot be used and hence this system as such can only perform during daylight (when marker/riser is visible in the camera). We are aware of this limitation and working on integrating ultrasonic sensor with our prototype to mitigate this shortcoming.
- Template matching technique is dependent on distance, angles and other ambient condition between subject and the camera. We need more discrete and variant image set to fully validate the scene-text recognition via template matching.

VIII. Conclusion

In this work, we have presented our vision on how to go about automating the flash flood detection via scene-text recognition technique. Though the setup was prototypical, we can lay a foundation for future expansion into this work. These preliminary experiments were performed at the same camera distance and ambient weather condition. The effect of distance and color segregation during various time of the day is yet to be tested. This work is still in progress, and there is a lot to be done in its entirety. We are paving our way to what we believe will be an innovative low-cost, scalable solution for flash flood detection and early warning system.

Acknowledgment

This work is partially supported by the NSF CNS grant 1640625.

References

[1] Alur R., Berger E., Drobnis A. W., Fix L., Fu K., Hager G. D., Zorn B. (2015). System Computing Challenges in the Internet of Things: A white paper prepared for the Computing Community Consortium committee of the Computing Research Association. http://cra.org/ccc/resources/ccc-led-whitepapers/

[2] K. Khurana and R. Awasthi, “Techniques for Object Recognition in Images and Multi-Object Detection,” International Journal of Advanced Research in Computer Engineering and Technology (IJARCET), vol. 2, no. 4, Apr. 2013.

[3] C. Shi, C. Wang, B. Xiao, Y. Zhang, S. Gao, and Z. Zhang, “Scene Text Recognition using Part-based Tree-structured Character Detection,” Computer Vision Foundation, 2013.

[4] R. Qian, B. Zhang, Y. Yue, and F. Coenen, “Traffic sign detection by template matching based on multi-level chain code histogram,” 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), 2015.

[5] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, “Reading Text in the Wild with Convolutional Neural Networks,” International Journal of Computer Vision, vol. 116, no. 1, pp. 1–20, Jul. 2015.

[6] ”What is template matching?” OpenCV 2.4.13.4 documentation. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials /imgproc/histograms/templatematching/template matching.html. [Accessed: 17/Dec/2017].

[7] [LeCun et al., 1998a] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. ”Gradient-based learning applied to document recognition.” Proceedings of the IEEE, 86(11):2278-2324, November 1998. [on-line version]

[8] K. C. Chang, P. K. Liu and Y. S. Wang, ”Parallel Design of Background Subtraction and Template Matching Modules for Image Objects Tracking System,” 2016 International Computer Symposium (ICS), Chiayi, 2016, pp. 18-21. doi: 10.1109/ICS.2016.0013

[9] MacQueen, J. Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics, 281–297, University of California Press, Berkeley, Calif., 1967.https://projecteuclid.org/euclid.bsmsp/1200512992

[10] Sonalkadu, Sheetal Dhande “Effective Data Mining Through Neural Network” , International Journal of Advanced Research in Computer Science and Software Engineering ,Volume 2, Issue 3, March 2012 ISSN: 2277 128X

[11] Nittel, S. A Survey of Geosensor Networks: Advances in Dynamic Environmental Monitoring. Sensors 2009, 9, 5664-5678. Nittel S. A Survey of Geosensor Networks: Advances in Dynamic Environmental Monitoring. Sensors. 2009; 9(7):5664-5678

[12] Maddox, “The Risks of Hazard,” Three Common Types of Flood Explained. [Online]. Available: http://www.intermap.com/risks-of-hazard-blog/three-common-types-of-flood-explained. [Accessed: 03-Feb-2018].

[13] US Department of Commerce, NOAA, National Weather Service. “Ellicott City Historic Rain and Flash Flood - July 30, 2016.” National Weather Service, NOAA’s National Weather Service, 1 Sept. 2016, www.weather.gov/lwx/EllicottCityFlood2016.[Accessed: 04-Feb-2018].

[14] Elizabeth M. Argyle, Jonathan J. Gourley, Chen Ling, Randa L. Shehab, Ziho Kang: Effects of display design on signal detection in flash flood forecasting. Int. J. Hum.-Comput. Stud. 99: 48-56 (2017)

[15] Tamari, S.; Guerrero-Meza, V. Flash Flood Monitoring with an Inclined Lidar Installed at a River Bank: Proof of Concept. Remote Sens. 2016, 8, 834.

[16] Dhafer Ben Arbia, Muhammad Mahtab Alam, Abdullah Kadri, Elyes Ben Hamida, Rabah Attia:Enhanced IoT-Based End-To-End Emergency and Disaster Relief System. J. Sensor and Actuator Networks 6(3): 19 (2017)

[17] L. Vojtech, M. Neruda, J. Skapa, J. Novotny, R. Bortel and T. Korinek, ”Design of RFID outdoor localization system: RFID locator for disaster management,” 2015 5th International Conference on the Internet of Things (IOT), Seoul, 2015, pp. 4-11.