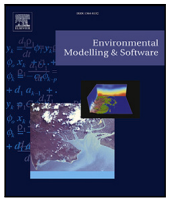
Environmental Modelling and Software 160 (2023) 105586

Contents lists available at ScienceDirect

Environmental Modelling and Software

journal homepage: www.elsevier.com/locate/envsoft



V-FloodNet: A video segmentation system for urban flood detection and quantification

Yongqing Liang [a], Xin Li [b,*], Brian Tsai [c], Qin Chen [d,e], Navid Jafari [f]

[a] Department of Computer Science, Texas A & M University, College Station, 77843, TX, USA
[b] Section of Visual Computing and Creative Media, School of Performance, Visualization, and Fine Arts, Texas A & M University, College Station, 77843, TX, USA
[c] Department of Computer Science, Louisiana State University, Baton Rouge, 70803, LA, USA
[d] Department of Civil and Environmental Engineering, Northeastern University, Boston, 02115, MA, USA
[e] Department of Marine and Environmental Sciences, Northeastern University, Boston, 02115, MA, USA
[f] Department of Civil and Environmental Engineering, Louisiana State University, Baton Rouge, 70803, LA, USA

ARTICLE INFO

Keywords:
Water level estimation
Water image/video segmentation
Object dimension estimation
Flood monitoring
Deep learning

ABSTRACT

Effective monitoring and forecasting of urban flooding are crucial for climate change adaptation and resilience around the world. We proposed a novel and automatic system for urban flood detection and quantification. Our software takes image/video data of flooding as inputs because the such data source is easy to obtain and widely available compared with conventional water level sensors or flood gauges. First, the kernel of our system is a robust water region segmentation module that detects flooded regions together with surrounding reference objects from the scene. We combine image and video segmentation technologies to make the system reliable under varying weather and illumination conditions. Second, our system uses the detected situated objects to determine the inundation depth. Field experiments demonstrate that our segmentation results are accurate and reliable; and our system can detect flooding and estimate inundation depths from images and time-lapse videos. Our code is available at https://github.com/xmlyqing00/V-FloodNet.

1. Introduction

Improving our ability to monitor and forecast flooding in urban environments is important to developing an early warning system. Flooding constitutes the largest portion of insured losses among all disasters in the world, accounting for 71 percent of the global natural hazard costs and affecting 3 billion people from 1995 to 2015 (Colgan, 2017; Aerts et al., 2014). More accurate urban flood mapping during a disaster will provide essential data on the number of damaged homes and businesses, which will assist urban centers in requesting and doling out emergency funding. It will also facilitate the estimation of the number of days businesses remained closed (*i.e.*, flood duration) and the amount of inventories lost, which will determine the economic impact and losses from the flood event. Real-time flood monitoring also helps with evacuation routing and first responders during disasters.

The ability to construct the rise and fall of water levels (*i.e.* hydrograph) in urban areas in real-time during hurricanes, fluvial and pluvial floods, and other extreme weather events is challenging because of the low spatio-temporal density of water level measurements and the complex interactions of built infrastructure and natural landscape with flowing water. Urban flood mapping commonly involves using incomplete information from sensors mounted on stream or tide gages, airborne sensing, satellite imagery, and physics-based flood models. Water-level gages are the most ubiquitous, but they are installed along water bodies far from urban areas to relay accurate and real-time data for emergency operation centers to make informed decisions. Airborne and aerial sensing imagery captures the spatial extent of flooding, but it lacks temporal resolution and estimates of flood depths. Physics-based models solve the spatial and temporal inconsistencies, but they require lengthy run-time duration and the prediction accuracy depends on the quality of the model inputs. Thus, an immediate need exists to find non-traditional data to complement and improve our flood prediction capabilities.

In this study, we use photos and videos that are widely available from various sources such as traffic cameras, webcams, social media, and so on, to detect flooding and estimate water inundation depths. Such kinds of images/videos are prevalent in many places in urban areas. As a result, with a software system developed upon such flexibly/widely distributed data sources, building a dense flood inundation map can become feasible.

* Corresponding author.
E-mail addresses: lyq@tamu.edu (Y. Liang), xinli@tamu.edu (X. Li), btsai2@lsu.edu (B. Tsai), q.chen@northeastern.edu (Q. Chen), njafari@lsu.edu (N. Jafari).

https://doi.org/10.1016/j.envsoft.2022.105586
Received 30 July 2022; Received in revised form 8 November 2022; Accepted 21 November 2022
Available online 24 November 2022
1364-8152/

Current approaches and limitations. Recent computer vision based systems for water depth estimation can be generally classified into two types. (1) The *first category of methods* directly uses observed reference objects to estimate water levels, without explicitly detecting or segmenting flood/water regions in the scene (Alizadeh et al., 2021; Kharazi and Behzadan, 2021; Meng et al., 2019; Park et al., 2021). The assumption in these methods is that incomplete reference objects are submerged in water. But if the reference objects are occluded by other objects or scenes (due to camera view angles), such an assumption is no longer valid and it leads to *less accurate prediction*. (2) The *second category of methods* explicitly segments both object and flood regions so that their boundary interface is identified and used for estimation (Geetha et al., 2017; Pally and Samadi, 2022; Chaudhary et al., 2019; Hu et al., 2017). Segmentation techniques used in these systems run in a frame-by-frame manner. However, water/flood has highly versatile and dynamic geometry and appearance under different motions, reflections, and weather/illumination conditions. Consequently, segmenting them frame-by-frame, without utilizing temporal and spatial coherence in the video often results in less stable and less accurate water segmentation and depth estimation.

Our approach. To achieve more accurate and robust water level estimation, we first develop a new video-based water segmentation framework to segment water and surrounding reference objects; and then with this segmentation, we design an inundation depth estimation algorithm to predict water depth. The whole pipeline consists of the following two steps. (1) The first step is *Water Region Segmentation*, which detects water/flood regions together with surrounding reference objects from the scene, and computes their boundaries and interface. An image-based segmentation model trained using water data we downloaded and labeled is used to detect and segment water from the first frame of the video. Then, a video-based segmentation model analyzes the change in the water region and tracks the water boundary in each of the subsequent frames. (2) The second step is *inundation depth estimation*. Detected reference objects that are incomplete and share a common contour boundary are used to estimate the water depth. Such a reference object is matched with its complete template model to calculate the dimensions of its visible and invisible (submerged) parts, where the vertical length of the invisible (submerged) part is reported as the water depth.

The **main contributions** of this study are summarized as follows.

- We built a new system to estimate water depth from images and videos. By integrating effective segmentation and dimension estimation techniques, this pipeline is automatic and can run robustly under different weather conditions.
- We designed a new deep learning segmentation pipeline that integrates both image-based and video-based segmentation models to detect and track flood and reference objects in long video sequences under varying weather conditions.
- We designed a new inundation depth estimation model based on template matching, and use it to automatically calculate water depth.
- We collected and annotated a comprehensive dataset that includes water-related images and videos, segmentation labels, and water depth information. Such a dataset is currently not publicly available. Our dataset will be useful for training water/flood detection and segmentation systems. We have released this dataset together with our open-source programs to the public.

2. Methods

We first review existing work in several topics closely related to this work (Section 2.1, and then introduce our proposed method (Sections 2.2∼2.6)

2.1. Related work

2.1.1. Water depth/level analysis

Recently, image/video based computer vision techniques have been used to estimate the depth or water level of floods. These methods can be generally classified into two categories.

One category avoids the challenging water segmentation task and directly estimates water depth using detected objects in the scene (Alizadeh et al., 2021; Kharazi and Behzadan, 2021; Meng et al., 2019; Park et al., 2021). These approaches assume that if an object is incomplete in images/videos, then the missing part is in the water. And the missing portion/ratio is directly used to calculate water depth/level. Ning et al. (2020) proposed a screening system to classify flood and non-flood related images from social media. This system can detect the existence of flood from the scene but cannot quantify the inundation depth. Recent papers detect common objects in the scene and estimate the water depth by computing the ratio of invisible parts to the whole structure. Commonly used objects include *stop signs* (Alizadeh et al., 2021; Kharazi and Behzadan, 2021), *human (skeleton)* (Meng et al., 2019), and *vehicles* (Park et al., 2021). These methods assume the invisible region of the reference object is submerged in water. But in many videos involving multiple objects or relatively complex scenes, this assumption does not hold and these models could yield unreliable estimations.

The second category first segments water regions in the scene and models their interface with submerged objects then performs water level estimation accordingly (Geetha et al., 2017; Pally and Samadi, 2022; Chaudhary et al., 2019). However, accurate water segmentation from images/videos is challenging. Water segmentation methods adopted in related estimation systems include *non-learning* based approaches (Geetha et al., 2017; Pally and Samadi, 2022), and *learning-based* ones (Chaudhary et al., 2019). *Non-learning based methods* such as binary thresholding (Geetha et al., 2017) and Canny edge detection (Pally and Samadi, 2022) are designed for relatively simple environments where the water region is a simple, largest connected component in the image. They have two major limitations. First, thresholds need to be manually adjusted and making the pipeline automatic and these parameters adaptive to practical scenes is very difficult. Second, water regions are often more complex than a simple and big connected component in practical images/videos, these systems could miss the detection of significant water regions. Recently, deep learning based technologies achieved state-of-the-art performance in image/video segmentation. Chaudhary et al. (2019) utilizes the Mask R-CNN (He et al., 2017) pipeline to segment water from the image. Erfani et al. (2022) proposes AQUANet to segment aquatic and non-aquatic regions from images. Mask R-CNN and AQUANet are image-based methods, and they run the segmentation frame-by-frame, without utilizing temporal coherency information in the video (see Fig. 10 for comparisons). Our observation is that *such temporal coherency information is critical, especially when segmenting appearance-volatile objects like water under dynamic weather and lighting conditions.* In addition, bounding box based segmentation such as Mask R-CNN could fail to correctly bound water regions due to their highly dynamic and deforming nature.

2.1.2. Object segmentation in images and videos

Effective segmentation of flood from input images and videos is critical for accurate segmentation-based flood estimation. We briefly review recent image and video segmentation techniques in computer vision.

Image segmentation. Recent use of deep learning based algorithms allows the complex network to robustly model both local and global characteristics of the interested objects/regions in images. A widely adopted deep learning based framework for semantic segmentation is the Fully Convolutional Network (FCN) (Long et al., 2015), which trains a series of convolutional layers to extract features and uses

a deconvolutional operation to upsample the feature vector to infer pixel-wise category.

Chaurasia and Culurciello (2017) proposed LinkNet, which connects feature extraction layers to their corresponding decoder layers to recover spatial information lost during encoder downsampling operations. Recently, three feature extraction architectures are popularly adopted: ResNet, Xception, and EfficientNet. ResNet stacks residual nets, special blocks of layers with skip connections to allow the bypassing of one or more layers (He et al., 2016). This allows for the creation of extremely deep networks while avoiding degradation in training accuracy. In contrast to the ResNet's focus on deep layers, Inception (Szegedy et al., 2015) and its improved model, Xception (Chollet, 2017), seek to discover optimal network width by applying multiple convolutional filters through Inception Blocks. Xception outperformed ResNet-152 and Inception V3 on the commonly used testing benchmark ImageNet. EfficientNet was developed to uniformly scale a network's width, depth, and resolution using a compound coefficient (Tan and Le, 2019). It reduces the necessity for manual tuning of individual network dimensions while using much fewer parameters compared to other convolutional networks of similar accuracy. More recently, EfficientNet used neural architecture search to optimize its network architecture and achieved state-of-the-art accuracy on the ImageNet benchmark. However, applying image-based segmentation in a frame-by-frame manner without considering temporal and spatial coherency between adjacent frames in the video could lead to inaccurate and less reliable segmentation.

Video segmentation. Recent video object segmentation techniques are mostly deep learning based, and they can be generally divided into two categories: *implicit learning* and *explicit learning* methods. The *implicit learning* approaches use masks computed in previous frames to infer masks in the current frame (Perazzi et al., 2017; Hu et al., 2017; Bao et al., 2018; Hu et al., 2018b). These methods often require online learning to adapt to new objects in the test video (where online learning means that the trained model requires additional training during inference time to fit the evaluation environment.) The *explicit learning* methods use previous frames to construct an embedding space to memorize the object appearance, then classify each pixel's label using their similarity (Oh et al., 2018; Voigtlaender et al., 2019; Wang et al., 2019; Lin et al., 2019; Hu et al., 2018a; Liang et al., 2020a; Oh et al., 2019).

However, popular video object segmentation methods face two key challenges that hinder segmentation performance in processing flood videos: (1) Most monitoring video are not short clips but are relatively long. Existing expensive frame-by-frame detection schemes could lead to an out-of-memory issue or significantly decrease the model performance (caused by temporal downsampling); and (2) The frequently changed appearance of the water makes reliable feature learning and tracking difficult. In this study, we extend our recently developed AFB-URR software pipeline, which organizes an adaptive feature bank (AFB) to flexibly and dynamically manage the features of objects and water in videos with changing weather and lighting conditions (Liang et al., 2020b). AFB-URR achieved state-of-the-art reliability and accuracy under such scenarios. Video segmentation methods require objects of interest in the first frame to be provided, often through a manual annotation. In this task, however, we need the entire system to run automatically. Hence, we propose to combine image and video segmentation technologies: using image segmentation to segment the first frame, and then using video segmentation to propagate the estimated water masks to the subsequent frames.

2.1.3. Deep learning models for hydrologic data

With the help of computer technologies, many systems have been developed recently to model and analyze rivers from images. For example, semantic segmentation models have been developed to identify and extract river, drainage, and hydrologic streamlines from images (Mao et al., 2021; Xu et al., 2021; Hosseiny, 2021). In these systems, the U-Net architectures, commonly used in image segmentation models, are adopted to extract features from images and then decode them to identify water regions. Ford et al. (2019) proposes a flooding assessment framework for climate change studies.

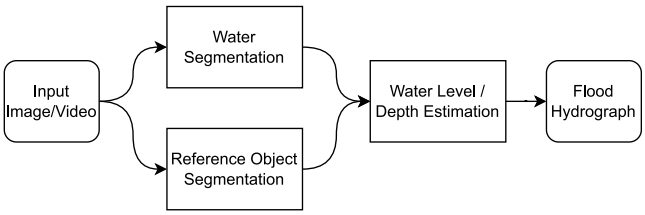


Fig. 1. An overview of the proposed V-FloodNet system. It takes a video (or an image) as input and estimates the water depth in two steps. First, the water region and reference objects are detected and segmented from the scene; their interface is also identified. Second, a template matching module aligns the detected object with its standard reference template, which is then used to rectify the view angles and poses, and estimate the flood depth.

2.2. Proposed approach: Overview

Our proposed V-FloodNet pipeline is shown in Fig. 1. It takes a video as input and performs both water and reference object segmentation. For water segmentation (Section 2.4), an *image based segmentation* module runs on the first frame (or any given reference frame) to detect and segment out the water region. With the *video based segmentation* module, this propagates to all the other frames. We also detect and segment reference objects from the scene (Section 2.5). After the contents in the video are segmented into the water, reference objects, and background regions, the interfaces between water regions and reference objects are detected. Then, a *water level estimation* module (Section 2.6) performs a template matching to align the detected reference objects with templates. This alignment normalizes (*i.e.*, rectifies the view of) the reference objects. It then superimposes the boundary interfaces onto the reference object to estimate water depth.

2.3. Water image data collection and annotation

To train an effective deep video analysis system, a large dataset of annotated image and video is needed. The well-known COCO Dataset (Lin et al., 2014) is a large public semantic segmentation dataset and benchmark, including many common scenes and objects. This dataset contains images and their pixel-wise annotations, and it has been widely used in many object detection programs. Most of the existing water estimation systems rely on the COCO dataset to train their object detection and segmentation models. However, COCO does not include water images, and cannot be used to train water detection and segmentation. In fact, annotated water/flood images are very limited on the internet. Erfani et al. (2022) propose an ATLANTIS Dataset that captures a wide range of water-related objects, but the number of images that have flood labels is limited.

Our previously collected WaterDataset (Liang et al., 2020a) includes general water images (like pool, lake, and ocean) from the ADE20K dataset (Zhou et al., 2017), and river images from the RiverSeg dataset (Lopez-Fuentes et al., 2017) where cameras are on the top of the water regions. To support model training in this study, we extended our previously collected dataset by adding flood images with pixel-wise annotations. We searched the images with the keywords "Flood Image" on the PixaBay website[1] and downloaded all related images (682 images). We manually selected 149 images that are good quality based

[1] https://pixabay.com/.

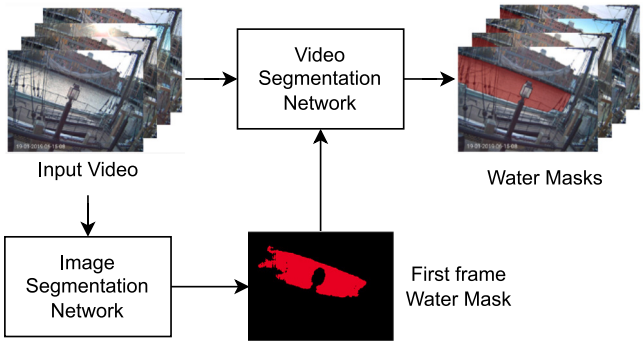


Fig. 2. The water region segmentation pipeline. It takes a video (or an image) as input. First, it uses the image segmentation module to estimate the water mask of the first frame. Second, it uses a video segmentation module to estimate the water masks of the rest frames.

on two criteria: (1) Images contain water regions in a moderate size, especially floods, and (2) Images are not over-retouching. Three people collaborated and manually annotated the selected images using a public open-source tool LabelMe (Wada, 2021). We divided the images with a certain overlap and assigned them to different people for annotation. After labeling, the consistency of annotations on the overlapped images was checked. We also re-examined all the images in WaterDataSet and removed those with tiny or no water regions. The final processed image dataset contains 1912 images from the ADE20K dataset (which includes general water-related scenes like pool, lake, and ocean), 300 river images from the RiverSeg dataset, and 149 flooding images from the PixaBay website.

As for the video data, we collected and labeled multiple videos with groundtruth water depth. One video recorded the rise and fall of the flooded levels in Houston Buffalo Bayou during Hurricane Harvey in 2017. A few other videos are from Boston Harbor, where a web camera is mounted at the Boston Harbor Tea Party Ship & Museum.[2] For these videos, we obtained the water level measurements from the Boston tide station (ID: 8443970, MA) of the National Oceanic and Atmospheric Administration (NOAA).[3] We also collected multiple flood videos from a creek on the LSU campus. We deployed a Raspberry Pi camera near this creek. It was controlled to capture and transmit the flood video (one image per minute) whenever there is a rain storm. A water level gage was installed at this location, which provides accurate reference depths.

Finally, the water dataset collected in this work includes 2361 images with pixel-wise annotations and 6 videos with accurate water level/depth groundtruths. Some example training images are shown in Fig. 3. We used this dataset for the training and evaluation of the proposed system. The dataset will be released in GitHub for comparative studies.

2.4. Water region segmentation

2.4.1. Image segmentation

Our image-based water segmentation framework relies on the popular encoder–decoder based architecture. The encoder is a neural network that encodes the input image into feature maps. Then the decoder takes the feature maps to predict the segmentation mask. We conducted experiments on three widely used feature encoders: ResNet-50 (He et al., 2016), EfficientNet-B4 (Tan and Le, 2019), and Xception (Chollet, 2017), and the state-of-the-art decoder network LinkNet (Chaurasia and Culurciello, 2017). All three feature encoders were pre-trained on the ImageNet dataset.

[2] https://www.bostonteapartyship.com/boston-webcam.
[3] https://tidesandcurrents.noaa.gov/waterlevels.html?id=8443970.

Through comprehensive experiments, the combination of EfficientNet-B4 and LinkNet achieves the best performance (see Section 3.1). Hence, we selected this combination as our image segmentation module.

2.4.2. Video segmentation

After the camera is deployed, it keeps capturing scene images and sending the videos for monitoring and analysis. The aforementioned *image segmentation model* will run on the first frame (or any given calibration frame) of the video. We can assume such a first/reference frame is taken in good weather/lighting conditions, whose segmentation is relatively easy. Then, in the next step, a *video segmentation model* will run to propagate this frame's segmentation to subsequent frames. In contrast to the direct frame-by-frame image segmentation approaches adopted in existing water analysis systems (Hu et al., 2017; Maninis et al., 2019), with this video segmentation strategy we can utilize the temporal and spatial coherency information across different frames. This is crucial to maintaining the segmentation robustness and accuracy under changing/severe weather and lighting conditions (*e.g.*, nighttime, rainy/foggy/snowy weather). Fig. 4(a–d) depicts several examples in which the water appearance significantly varied owing to the changes in weather/lighting conditions. Fig. 2 shows the pipeline of the proposed water segmentation model.

The proposed video water segmentation module is built upon our recently developed architecture AFB-URR (Liang et al., 2020b). AFB-URR automatically adjusts the learned features to capture the transition of the object's appearance changes. This allows the system to more robustly propagate segmentation (masks) of water or other appearance-volatile objects to subsequent frames in long videos. We recap the key designs of the video water segmentation here. We formulate the segmentation problem as a pixel-wise classification task that labels each pixel as objects of interest (water, or other reference objects) or background. The video segmentation module consists of two components, *adaptive feature bank*, and *segmentation network*.

Adaptive Feature Bank. We maintain two adaptive feature banks (AFB) F_{obj} and $F_{background}$ separately to store the water and non-water features. Once we segment a new frame, we compare its features with the feature banks, then the similarity scores are decoded into the water mask. To adapt the object appearance changes in the video, we update the feature banks by three operations, *merging*, *appending*, and *deleting*. When a new feature is distinct from the existing features, we append the new feature to the feature banks. In contrast, when a new feature f_{new} is close to the existing ones in the feature bank, we merge them by updating the closest feature descriptor $f_{closest}$ from the feature bank by weight averaging,

$$f'_{closest} = \alpha f_{closest} + (1-\alpha) f_{new}, \tag{1}$$

where $f'_{closest}$ is the updated feature descriptor and we set $\alpha = 10$ in our experiments. The feature bank discards obsolete features according to the least frequently used (LFU) index. During the video segmentation process, such feature banks allow our segmentation system to effectively memorize the temporally changing appearance of foreground/background objects, as the feature banks adaptively absorb/adjust feature descriptors according to the changing scene.

Segmentation Network. The segmentation network also follows the encoder–decoder architecture. The *Encoder* is designed to encode the current frame I_t at frame t to its feature map f_t for segmentation.

$$f_t = Encoder(I_t), \tag{2}$$

where $I_t \in \mathcal{R}^{h \times w \times 3}$ is an RGB image with height h and width w. We build an attention module *Atten* (Oh et al., 2019) to compute the similarity between the features of current frame f_t and the feature banks F_{obj} and $F_{background}$. The attention module is used to make the network learn and focus more on the important information rather than

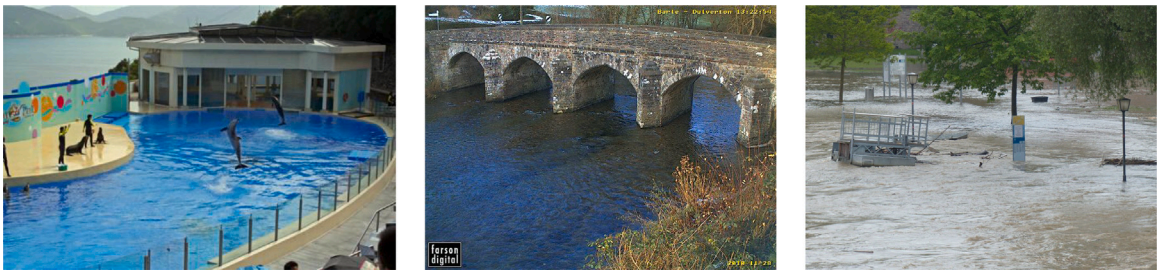


Fig. 3. Example images of the training dataset. We collected 2261 water-relevant images from three datasets for training: 1912 general water images (pool, lake, ocean) from the ADE20K dataset (Zhou et al., 2017), 300 river images from the RiverSeg dataset (Lopez-Fuentes et al., 2017), and 149 flooding images from the PixaBay website (https://pixabay.com/).

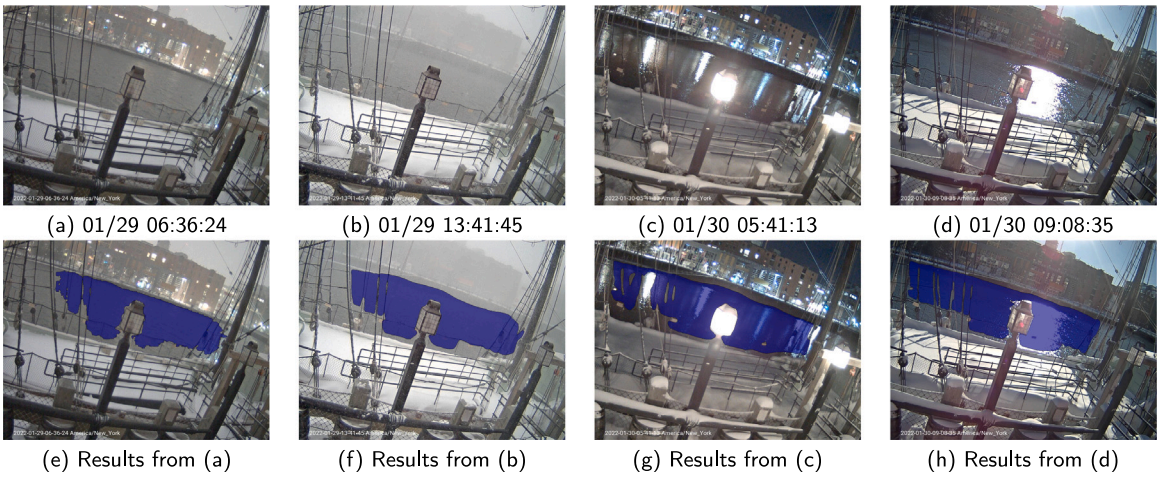


Fig. 4. Example frames from a video clip in January 2022, the camera is located at the Boston Harbor Tea Party Ships & Museum. The water's appearance changes on the same day under different conditions: (a) in the morning, (b) snowy, (c) in the evening, and (d) sunny. More details are discussed in the field study. In (e–h), blue masks indicate the segmented water regions.

non-useful features. We use this attention module to compute the most relevant features g_{obj} and $g_{background}$ from the feature banks:

$$\begin{aligned} g_{obj} &= Atten(f_t, F_{obj}), \\ g_{background} &= Atten(f_t, F_{background}). \end{aligned} \tag{3}$$

Then, a *Decoder* network takes the feature maps of the current frame f_t and combines it with similarity information calculated from the attention module to estimate the water segmentation mask S_t for the current frame. In the decoder module, we gradually upscale the size of the g_{obj} and $g_{background}$ by using the information from feature map f_t. We assign the class label for each pixel as the estimated segmentation mask S_t by checking the largest score from object or background maps,

$$S_t = Decoder(f_t, g_{obj}, g_{background}), \tag{4}$$

where S_t has the same resolution as the original input frame and it will be used to update the feature bank. The widely-adopted cross-entropy loss function is used to train the video segmentation model. In Fig. 4(b–c), we selected challenging frames where water appearances are affected by different weather and illumination. Fig. 4(a) is the first frame of the video, we used the image segmentation module to segment it in Fig. 4(e). The segmented water regions are marked in blue. Our video segmentation module built two feature banks F_{obj} and $F_{background}$ to memorize the appearances of the object (water) and background. Then, these two feature banks were used to segment the following frames. Meanwhile, the estimated water masks continuously update the feature banks. The feature banks adjust their learned features by analyzing the transitions of new features to adapt to the changes in appearance. Fig. 4(e–h) show the segmentation results of the Boston Harbor video. We can see that our video segmentation system has the ability to produce accurate and robust segmentation of flood water that has rapidly changing appearances.

2.5. Reference object segmentation

To estimate the inundation depth from the scene, we detect and segment reference objects that are submerged in water. Then we use estimated water region and reference objects to compute the submerging interface. We adopted the Mask R-CNN (He et al., 2017) network for object segmentation. We used the model pretrained on the COCO dataset (Lin et al., 2014) which can segment 80 classes of common objects, including cars, bikes, people, stop signs, and so on. We can also use the new video segmentation pipeline introduced in Section 2.4.2 to segment reference objects. In our experiments and release programs, we simply used the Mask R-CNN pipeline because of two main reasons: (1) We noticed that most common reference objects have the relatively stable appearance (*e.g.*, objects such as human beings, stop-signs, and buildings that we incorporated in this work) so it is often not necessary to maintain adaptive feature banks for reference objects; and (2) Unlike water, the detection and segmentation models trained on the big COCO dataset are robust and effective.

Once the reference objects are segmented, we extract a parameterized structural representation for each reference object using a *standard template* of this object. For example, in this work, we model a stop sign template in 2D as an octagon (plate) of 8 vertices and a straight line (pole) of 2 ending vertices (Fig. 11(c)), and model a person using a parameterized triangular mesh model (Fig. 12(c)). For other

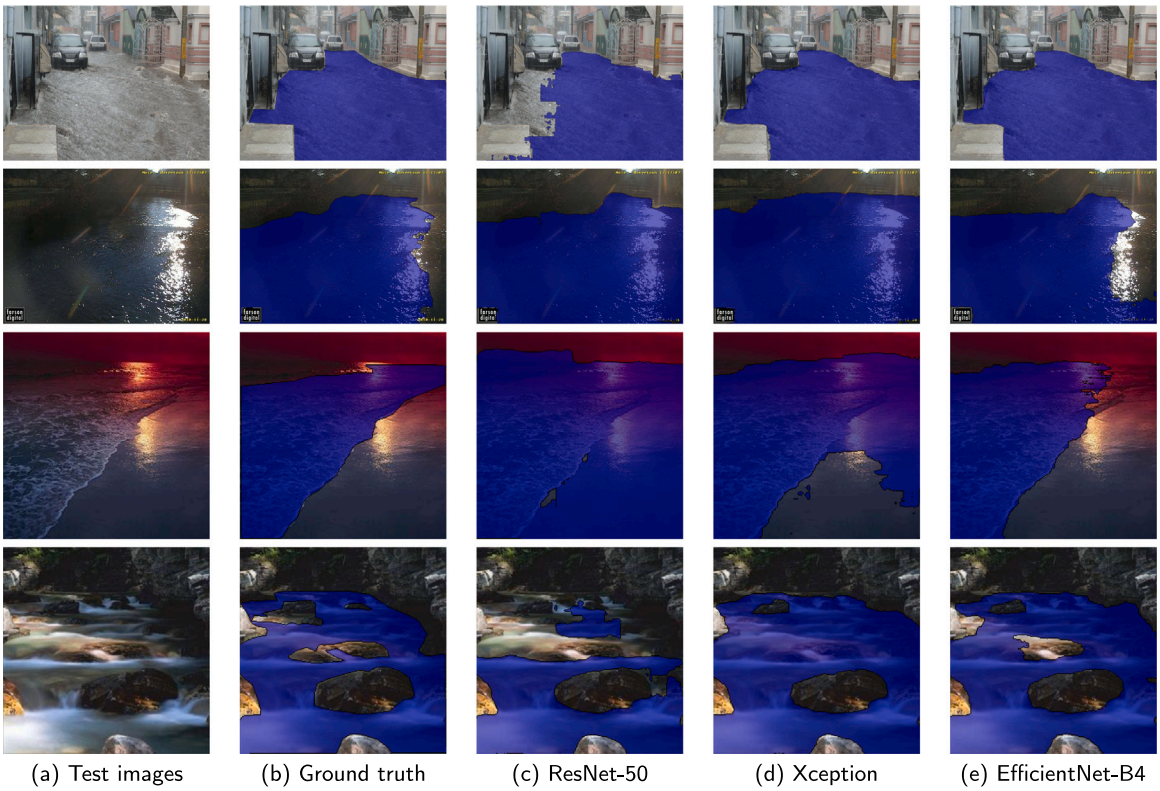


Fig. 5. Sample image segmentation results from the collected *WaterDataset*. We compared three widely-used feature encoders: ResNet-50, Xception and EfficientNet-B4.

general objects, if we can define a standard structural representation as its template, we can similarly model that object by fitting it with its template. Such a parametric template is in general modeled as a graph consisting of vertices (with coordinates) and edges (connecting vertices), and it is then used for dimension estimation.

Reference objects are often incomplete, with some parts invisible due to submergence or self-occlusion. To recover the missing part of an object, we match the object structure with its template structure on the visible parts. Once we know the visible (above water) and invisible (underwater) parts of the object structure, we can compute the water boundary and estimate the water depth in the scene. In this work, we tested our design on two types of reference objects: stop signs and human beings, which are often seen in flood scenes. Our template-based estimation scheme can be readily generalized to other reference objects, as long as their standard dimensions are known and their templates can be modeled. In the following, we elaborate on how we model and match the templates of stop signs and people, and use them to perform water depth estimation.

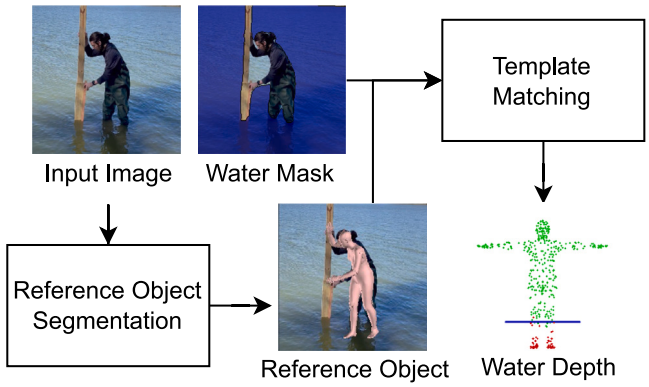


Fig. 6. The water depth/level estimation pipeline. It takes an image as input. First, it uses reference object segmentation to segment the shape of the reference object from the frame. Second, it uses template matching to register the estimated object shape to the template. Third, we compute the submergence ratio to estimate the water depth in the scene.

2.6. Water depth/level estimation

Built upon the water and reference object segmentation, our water depth estimation algorithm is automatic and it has two steps: *Template Matching* and *Submergence Ratio Calculation*. Fig. 6 shows the pipeline of the proposed water depth/level estimation system.

Template Matching. Given an estimated reference object M_O and a standard template (of this type of object) M_T, we match M_O and M_T by finding a spatial transformation that minimizes vertex displacements,

$$T = \arg\min_{T} \frac{1}{N} \sum_{i=1}^{N} \|T(v_t(i)), v_o(i)\|_p, \tag{5}$$

where $v_t(i) \in M_T$ and $v_o(i) \in M_O$, and i iterates the N vertices. T is the transformation function that aligns the vertices from the template structure to the object structure. The feasible space of T can be defined based on prior knowledge or physical properties of the reference objects or based on appropriate regularization schemes. For example, stop signs are planar and rigid, and are unlikely to undergo free-form deformations. Hence, we can restrict the feasible space of T to be rigid transformations when matching a stop sign with its template. When using human beings as reference objects, they are often modeled with more complex 3D geometric models which can deform more flexibly in a non-rigid manner. Their feasible space should contain more general free-form deformations. More specifically, here we adopt a 3D template human model (Lin et al., 2021) that contains 431 vertices, and allows T to be free-form deformation and regress it using a neural network.

Then, from its feasible space, we search for a transformation T that best aligns the reference object with its template. With (the inverse) T, we can transform and superimpose the template onto the reference

object in the scene. This allows us to obtain a "standardized" reference object $M'_T = T(M_T)$. Fig. 12(c) shows such an example on a reference person, where a template mesh (in peach color) is transformed to superimpose the detected reference object in the image scene.

For rigid and planar objects, such as stop signs, the transformations in different images can be related with a Homography matrix (Hartley and Zisserman, 2003). Hence, we solve a homography matrix with 8 degrees of freedom by matching the detected octagon (stop sign plate) and the template octagon. Similarly, the transformed stop sign template can be computed (see Fig. 11).

Submergence Ratio Calculation. Template matching allows the system to deform the template to superimpose the reference object in the scene. The water segmentation mask separates this template (its vertices/nodes) into two parts: the emergent part V_a, and the submerged part V_s. Based on the assumption that the water level is locally horizontal, our system computes the submergence ratio on the template using a vertical length ratio of V_s over $V_s + V_a$. We use h to represent the heights on the vertical axis, and the submergence ratio r_{water} is computed by

$$r_{water} = \frac{h(V_s)}{h(V_s) + h(V_a)}. \tag{6}$$

To estimate actual water depth d_{water} from the submergence ratio r_{water}, this ratio should multiply the physical size of the reference object l_{obj}, then add the vertical distance from the bottom of the reference object to the ground b,

$$d_{water} = r_{water} * l_{obj} + b, \tag{7}$$

where l_{obj} is the vertical dimension of the reference object. In practice, we often set l_{obj} using the dimension of the template, specifically the standard height of this type of reference object.

Estimation without Reference Objects. When no reference object is detected in the scene, our system allows the user to pick an object as a reference and give an estimated dimension. The reference object or region is supposed to be vertically above the water. Then our system will automatically track the reference object in all frames and compute the vertical distance from the reference object to the ground. The estimated water mask separates the vertical line into two parts: the line segment that is emergent (visible, denoted as V_a) and submerged by water (invisible, denoted as V_s). Similarly, we compute the submergence ratio r by Eq. (6), and convert it to the estimated water depth by Eq. (7). Our system also supports choosing multiple objects as reference objects. The estimated water level is the average of each reference object.

3. Results

We evaluated each sub-component of our water estimation system and tested the entire pipeline on videos captured in the field. Our code and dataset are available at https://github.com/xmlyqing00/V-FloodNet.

3.1. Evaluations on image segmentation

To select the best image segmentation model, we conducted experiments on three recent encoder–decoder based architectures: ResNet50-LinkNet (He et al., 2016; Chaurasia and Culurciello, 2017), EfficientNetB4-LinkNet (Yang et al., 2018), and Xception-LinkNet (Chollet, 2017). We trained these models using our WaterDataset, where images are randomly divided into 80% (training), 13% (validation), and 7% (testing), respectively. We provided the training details in Section 4.1. After each epoch, the models are evaluated on the validation images.

The experiment results are summarized in Table 1. The parameter size measures the number of learnable parameters in the network and the unit is million (M). Sample segmentation results are shown in Fig. 5.

Table 1
Image segmentation results on the test set of the proposed *Water Dataset*. We select the combination of EfficientNet-B4 and LinkNet as our image segmentation model. The parameter size measures the number of learnable parameters in the network and the unit is million (M).

Architecture	mIoU score	Parameter size
ResNet50 - LinkNet	0.7063	31M
Xception - LinkNet	0.7356	27M
EfficientNetB4 - LinkNet	0.7619	18M

Because of its best mIoU score (0.7619) on water segmentation and the smallest model size (parameter size 18M), we picked the combination of EfficientNet-B4 and LinkNet as our image segmentation model.

Comparison with existing water segmentation systems. We compared our image segmentation module with the existing water segmentation papers (Geetha et al., 2017; Chaudhary et al., 2019; Pally and Samadi, 2022). We directly used our trained model to segment the images from their paper without any additional training or tuning. Figs. 7, 8, and 9 show the water segmentation result comparisons. Segmentation from these existing papers tends to be incomplete or produce mis-classified foreground/background. In contrast, our water segmentation model can more reliably and accurately identify water regions from images. Our model was never trained on these images, which demonstrates the generalization of our segmentation model on flood images from new scenes.

3.2. Evaluations on video segmentation

A key advantage of our proposed system is that it can leverage the temporal coherency in the video to further enhance the reliability of water segmentation in videos. This is critical when it is needed to monitor and analyze scenes during both day and night time, and/or under both clear and inclement weather conditions. When our system takes video as input, it first uses the image segmentation model to segment water/flood and reference objects in the first frame, then our video segmentation model propagates the segmentation of water and other reference objects to subsequent frames utilizing not only texture/appearance but also spatial–temporal coherence in the video. Fig. 4 shows several example frames from a video recorded at the Boston Harbor. This video spanning several days covers different weather/lighting conditions, including (a) clear, (b) snowy, (c) dark, and (d) sunny weather conditions. On a relatively clear day, our image segmentation can detect the precise water boundary (*e.g.*, segmentation for the first frame), but as the lighting and weather conditions change, the water and background appearances can change significantly. In many scenarios (*e.g.*, nighttime, inclement weather, windy and rainy days, etc.), the image-based water segmentation accuracy drops significantly, because not only it is difficult to include all the possible scenarios in training datasets but also it is hard to memorize all possible water appearances caused by ripples, reflections, etc. By using spatial–temporal consistency exploited from the video, our video segmentation technique can greatly improve water segmentation robustness and accuracy. Fig. 10 compares the segmentation results of AUQANet (Erfani et al., 2022) and our models in some scenarios. Column (a) shows two frames, one taken on a snowy and foggy day (top), and one in the nighttime (bottom) (the reflections and lighting make the water looks drastically different from the daytime). When the image segmentation model is directly applied, the results (the second column) are less desirable (see columns (b) and (c)). But when the whole video sequence is considered, our video segmentation model gradually and adaptively adjusts the water features, and eventually produce much better segmentation results (column (d)) in both scenarios.

Fig. 7. Image segmentation comparison with (Geetha et al., 2017). Input images are from their papers. The green contours are from their segmentation. Our water segmentation (right column) is marked in blue.

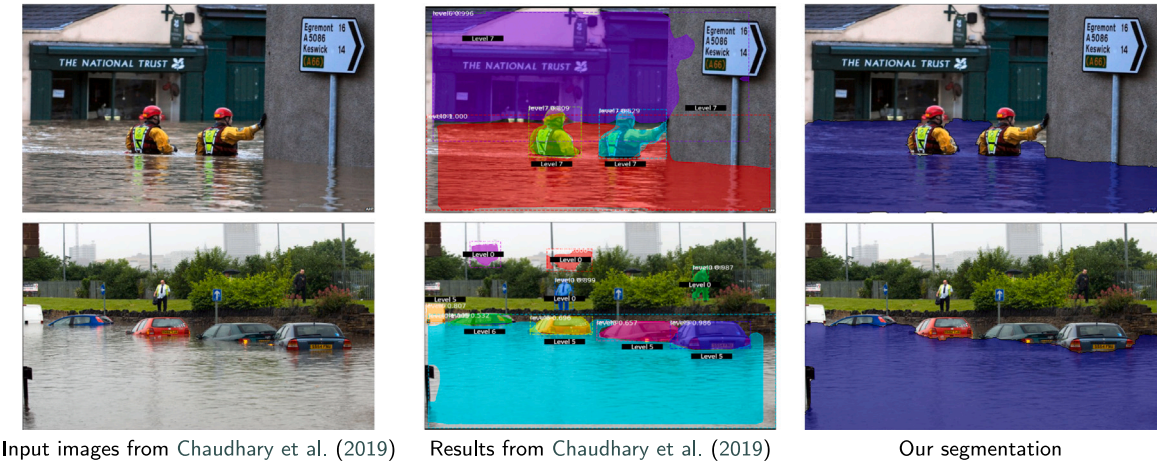


Fig. 8. Image segmentation comparisons with (Chaudhary et al., 2019). Input images are from their papers. The middle column shows their segmentation: boundary regions are often incomplete. Our water segmentation (right column) is marked in blue.

3.3. Inundation depth estimation

We conducted multiple field experiments to evaluate our flood water and inundation depth estimation pipeline. In the first field experiment, we deployed a standard stop sign in a shallow lake on the Louisiana State University (LSU) campus in Baton Rouge, Louisiana, USA. Next, we further tested our system with multiple monitoring videos of actual flooding or water level changes recorded during storms or severe weather conditions. These videos were from (1) the LSU campus creek, (2) Boston Harbor, and (3) Houston Buffalo Bayou. The two videos at the creek on the LSU campus were captured during two heavy storms in the summer of 2020. The three videos at the Boston Harbor were recorded several days during winter and summer with different weather/lighting conditions. The video at the Houston Buffalo Bayou recorded the water level changes during Hurricane Harvey (2017).

Field Experiment 1: LSU Lake.

Fig. 11(a) shows a frame in a video: a student stands in the water holding a standard stop sign. A ruler is attached to this stop sign to measure the actual water depth. At this location, the water depth is 0.343 m (see Fig. 11(f)).

The automatic water region segmentation from our system is shown in the blue mask in Fig. 11(b). We show that our system can use either the stop sign or the person to estimate the water depth. When using the stop sign as the reference, the system matches a template model (Fig. 11(c)) to the detected object in the image. Then, the visible region (Fig. 11(d)), superimposed with the water-object interface, infers the submergence ratio r_{water} on the stop sign template (Fig. 11(e)). Following U.S. Department of Transportation (2003)'s report, a standard stop sign on a conventional road in the USA has a 0.79×0.79 m plate and a 2.159 m long pole, where 0.75 m (30 in.) across opposite flats of the red octagon with a 0.02 m (3/4 in.) white border. With Eqs. (6) and (7), and this standard dimension ($l_{obj} = 2.159$ m and $b = 0$), our system estimated the average water depth at this location to be 0.359 m. Hence, with the stop sign as a reference, the prediction error is 0.016 m (4.7%).

When using the person as the reference, the 3D template mesh is deformed to fit the detected person in the image (Fig. 12(c)). Similarly,

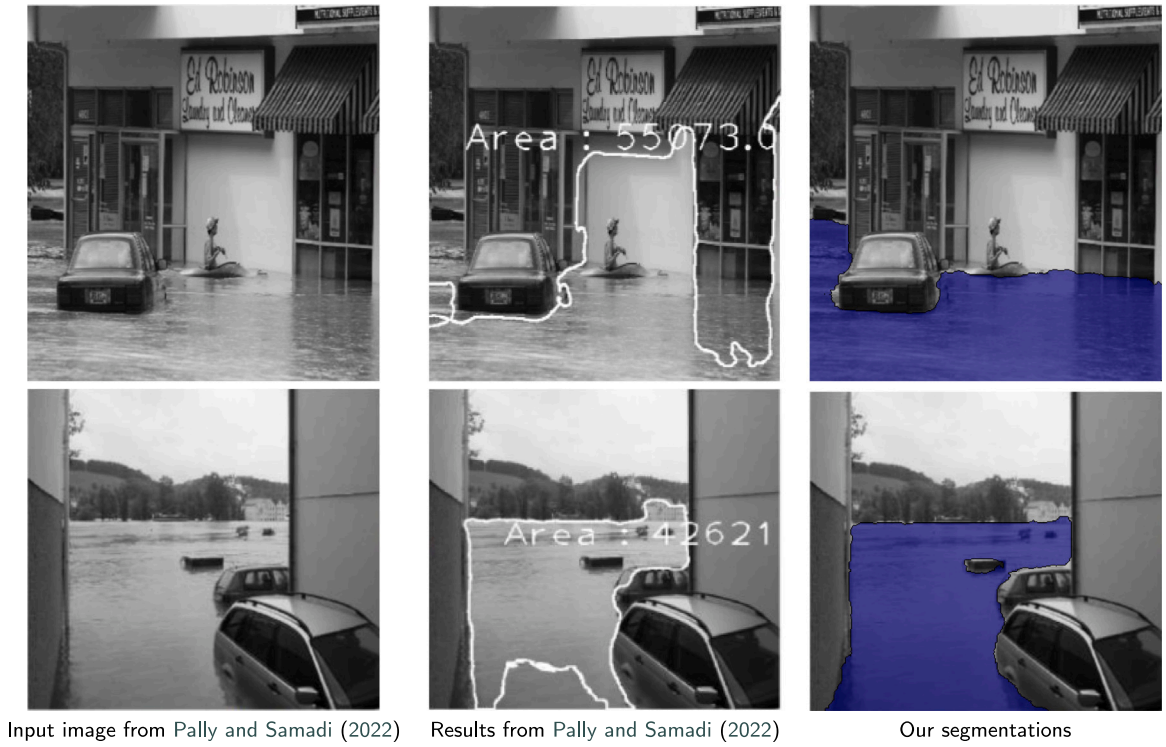


Fig. 9. Image segmentation comparisons with (Pally and Samadi, 2022). Input images are from their papers. The middle column shows their segmentation: background with a similar texture is often mis-classified. Our water segmentation (right column) is marked in blue.

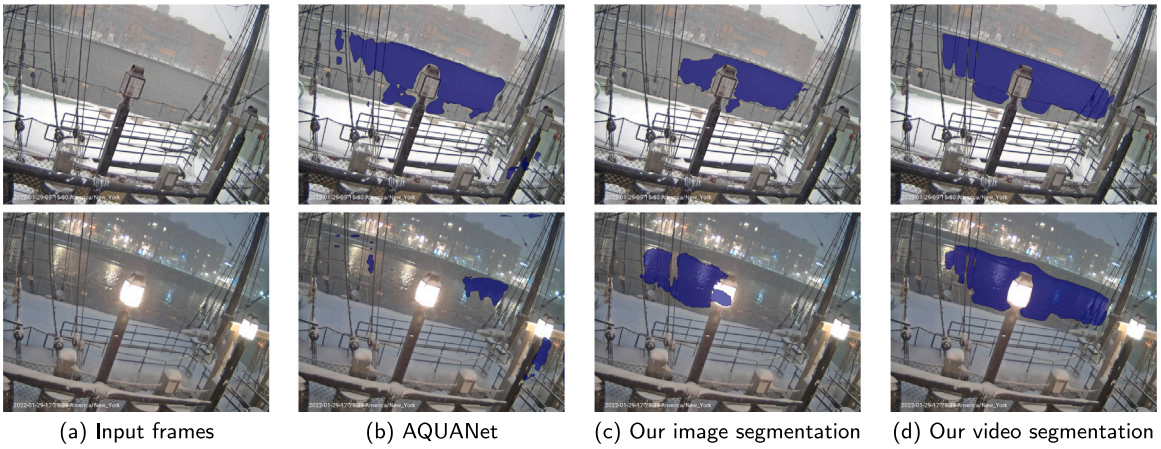


Fig. 10. Comparisons among AQUANet (Erfani et al., 2022), our image segmentation model, and our video segmentation model. (a) show two challenging moments from Boston Harbor Video (recorded on 2022/01/29). (b) show the segmentation results from AQUANet. (c) and (d) show the segmentation results by our image and video models. Our video segmentation model can segment consistent water regions by using temporal information to adapt to the water appearance changes.

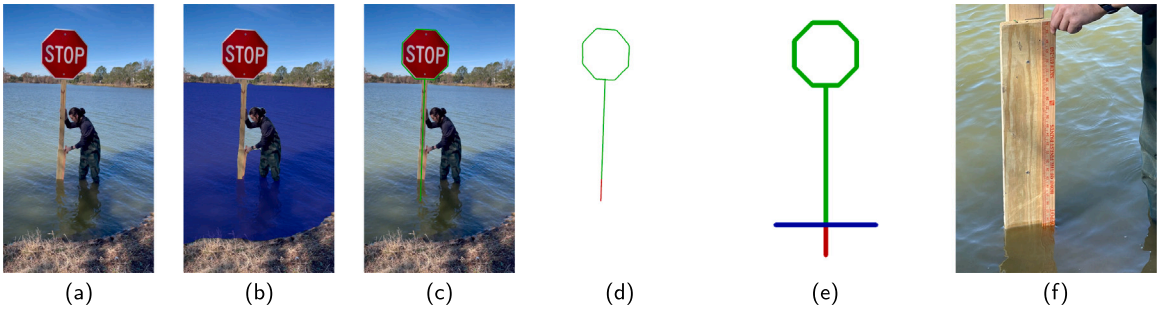


Fig. 11. Using stop signs to estimate water depth. When a stop sign is detected in the given image (a), our V-FloodNet segments the water region (as shown in the blue mask in (b)) and the stop sign (c). The reference stop sign is then matched (d) with a template model. And using its interface with the water region (e), the estimator module calculates the submergence ratio. In this experiment, the stop sign is equipped with a marking gauge, which provides the groundtruth water depth (f).

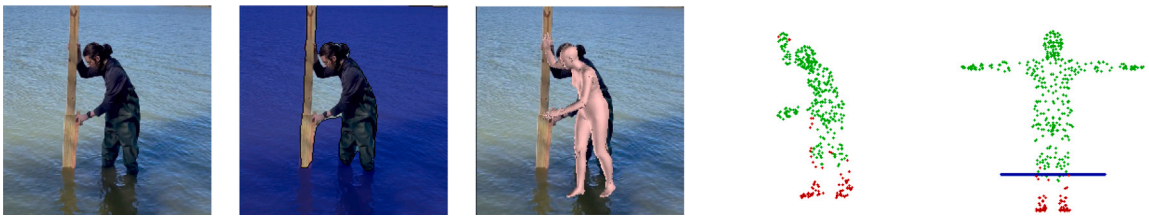


Fig. 12. Using human beings to estimate water depth. A detected person (a) can also be used as a reference to estimate water depth. The segmentation of the person and the water region (b) is used to find their interface. A template (for non-rigid and non-planar reference objects like human beings, we adopted a parametric 3D mesh as its template) is fitted and spatially superimposed (c). The boundary interface infers the submerged vs visible part and their size ratio (d), which is transformed back to the template coordinate system to estimate the water depth (e).

with the superimposition of template mesh and the water interface, the above-water compared to submerged parts of the human body are computed (visualized in green and red points respectively in Fig. 12(d)). The average submergence ratio on the person is then calculated to be 0.1882 (Fig. 12(e)). Finally, using the average height of the adult male (1.754 m, following Fryar et al., 2018), the system estimated the average water depth at this location to be $0.1882 \times 1.754 = 0.330$ m by Eq. (7), where $l_{obj} = 1.754$ and $b = 0$ because the adult male stands on the ground. Hence, using the person as a reference, the prediction error is 0.013 m (3.8%).

In our current system, we use the standard dimension of the reference objects as a default dimension. If the exact dimension of the reference object is known/different, we can simply adjust the parameters in the calculation to obtain a more accurate estimation.

Field Experiment 2: LSU campus creek

We deployed a Raspberry Pi Camera near the LSU creek to continuously capture the water depth change of the creek. We used one video that captured a rain storm on June 24, 2020 as an example to demonstrate our experiment. Fig. 13(a) shows one frame of this video. Fig. 13(b) illustrates the segmented water region (in the blue mask). Because there is no known reference object detected in this scene, the systems asked the user to pick a landmark and the program used it as the reference to calibrate the scene. In this experiment, the top of the gauge was selected as the reference (marked in the green box at the top of the gage in Fig. 13(b)). A simple vertical line from the landmark to the ground/water interface is used to estimate water depth. The system partitions this reference vertical line into the above-water and under-water parts, separated by the water interface. The submergence ratio r_{water} is the ratio between the dimension of underwater part and that of the entire line. We convert the submergence ratio to water depth by Eq. (7). When the camera was first deployed, its position, orientation, and focal lengths were calibrated, and the object size l_{obj} and b were calculated and fixed. After this step, the system can automatically estimate the water depth over time. The results are plotted in Fig. 13(c). The average absolute error of the estimated water depth in this experiment is 0.018 m and its standard deviation is 0.012 m (see Table 2).

Field Experiment 3: Boston Harbor

We tested our system on several long videos to demonstrate its robustness in varying weather and illumination conditions. In our collected WaterDataset, there are three long videos recorded on the Boston Harbor. The camera is mounted on the Tea Party Museum Ship and we downloaded the video one frame per ten minutes for multiple days. Fig. 14(a) shows one frame of the recorded video (from 2019/01/18 to 2019/01/23). When no salient reference objects are detected in the video, our system allows users to adopt one of two interactive mechanisms for calibration: (1) User can select four points to represent the horizontal axis and vertical axes, then the system can use them to estimate a perspective transformation to calibrate the scene; and (2) User can pick an arbitrary region in the scene as the reference. In this experiment, adopting the second mechanism, a region on the background building was selected and tracked. Then, pixel scale in the videos can be converted into physical units by fitting the parameters l_{obj} and b in Eq. (7).

Tracking water levels in these videos is highly challenging, due to two main reasons: (1) Different weather and different day/night illumination make the water appearance change continuously and significantly; and (2) The water regions are often occluded by other moving objects (*e.g.*, ships, people). Existing image and video segmentation systems cannot effectively detect and segment water regions in such situations. Our results are reported in Table 2, where both the segmentation and water level estimation are reliable and produce small errors. More visualization results are attached in the supplementary video on our GitHub page.

Field Experiment 4: Houston Buffalo Bayou

Hurricane Harvey (2017) made landfall along the Texas coast on 25 August 2017 as a Category 4 hurricane. This major storm caused catastrophic flooding of the densely-populated regions of Houston and Beaumont, leading to flooding of major interstate highways, such as I-10 and I-45. A high-water mark on the northern bank of Buffalo Bayou revealed the maximum flood water elevation of 10.3 m (33.7 ft) NAVD88 at the study site. Details about the flooding can be found in Jafari et al. (2021). At Milam Street, the rising floodwater reached an elevation of 8.3 m (27.1 ft) on August 27, 2017, a water level of 11.3 m (33.85 ft) above the bottom of the bayou (Harris Country Flood Warning System, 2017). However, the stream gage at Milam Street only collected data until 02:44 on August 27. This gage failure demonstrates the need for multiple methods to construct and verify flood hydrographs.

During Hurricane Harvey, a time-lapse camera was placed on the second floor of the Bayou Place Offices building on Capitol Street near Milam Street and overlooked Buffalo Bayou, Memorial Drive overpass, Interstate 45 overpass, and the Houston Aquarium. The camera recorded a video of the rise and fall of flood levels in Buffalo Bayou from August 25 to 31. Our system took the video as input to estimate the water level and then created the hydrograph. Our system used two bridge piers adjacent to Buffalo Bayou as reference objects to estimate the changes in the water level over time. We used the same conversion function (Eq. (7)) to convert the estimated submergence ratio to the water levels. The elevation of the bottom of the pier was found to be 0.83 m (2.72 ft) NAVD88 and the top 18.11 m (59.41 ft) NAVD88. After calibration, we have $l_{obj} = 18.11 - 0.83 = 17.28$ and $b = 0.83$. The system can then estimate the water level following Eq. (7).

With the water level estimated by our system, we compared it with the Milam Street gauge (the closest measure information available near Buffalo Bayou) and the results computed from our previous model (Jafari et al., 2021). The comparison is plotted in Fig. 15(c). In the morning of August 26, the Milam stream gauge shows an initial rise of the hydrograph. At midnight of August 27, the Milam Street gauge failed. Therefore, the blue curve stops. Our proposed model successfully captures the water rise on the morning of August 26. In contrast, the previous model (Jafari et al., 2021) could not catch this

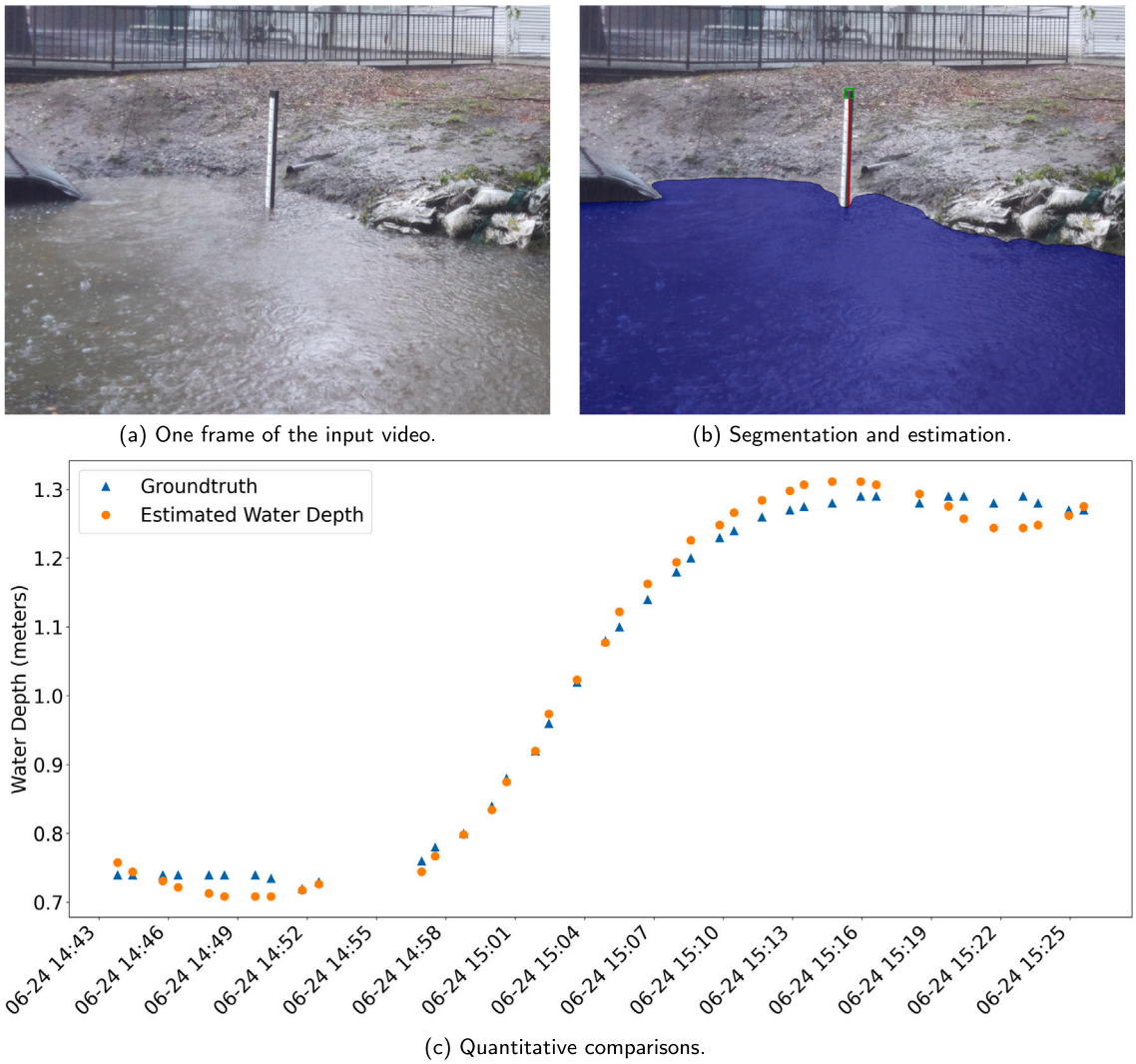


Fig. 13. Field experiment on heavy rainy days at LSU campus creek. (a) shows the original image and (b) shows the water segmentation (blue mask), reference marker (green box), and the estimated water depth (red line). (c) plots the estimated water depth and the groundtruth.

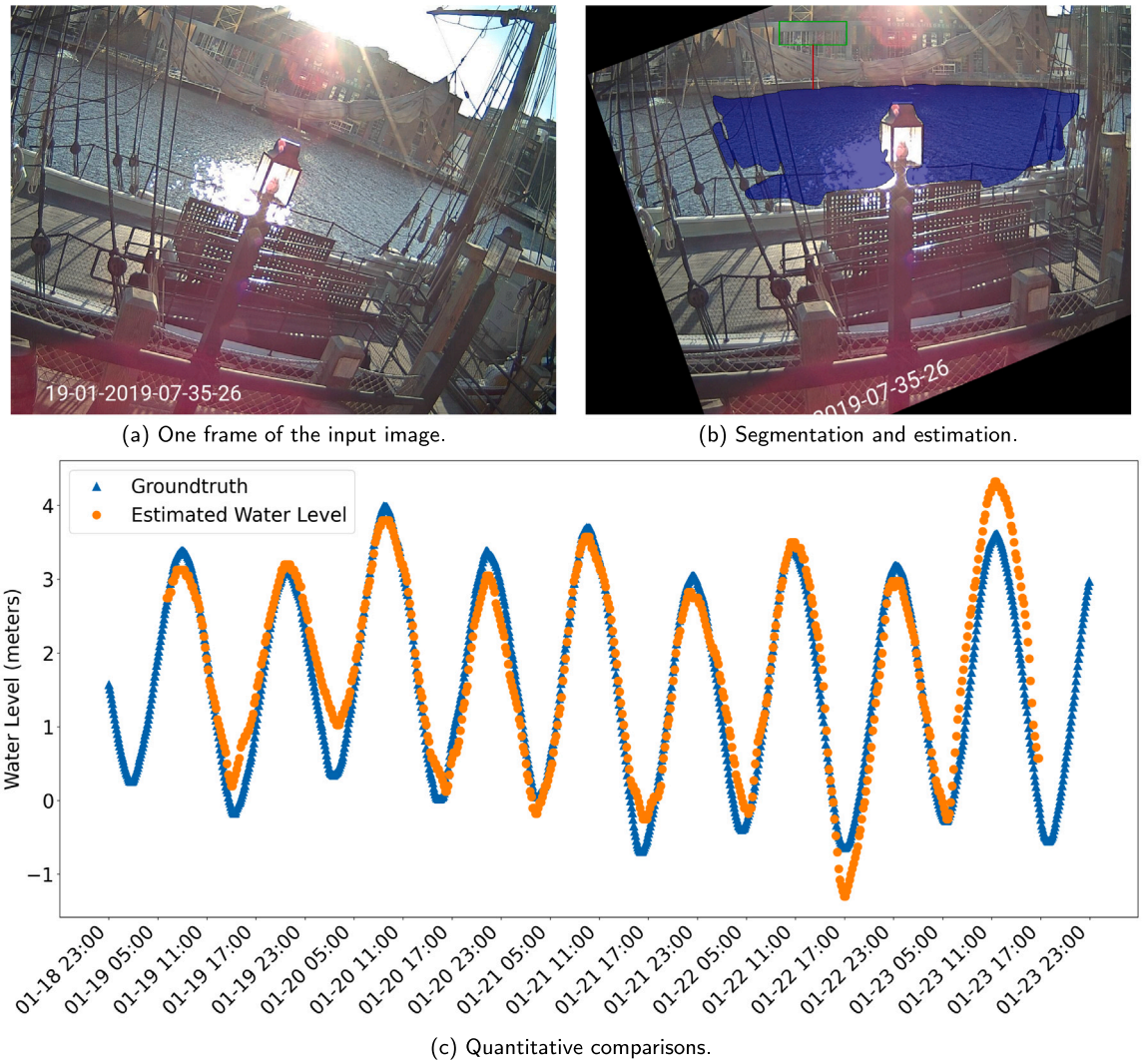
rise effectively. After the Milam Street gauge failed on August 27, our proposed model can still provide information on the flood. Specifically, according to U.S. Geological Survey (2017) (USGS), a high water mark[4] recorded the peak flood level of approximately 10.3 m occurred in the early afternoon of August 27. The hydrograph reconstructed by our V-FloodNet reaches the peak flood elevation at a similar time. By contrast, the previous model (Jafari et al., 2021) estimated the maximum water level over 14 m, which is significantly higher than the observed high water mark reported by USGS. The main reason that the previous model failed to produce accurate estimation is that it uses an image-based segmentation model to identify the water boundary. Under severe and changing weather and illumination conditions, image-based models cannot reliably stably track the water region. Our proposed model uses a video-based model, which leverages the spatial–temporal coherence in the video to achieve a more robust estimation. It is important to note that during Hurricane Harvey, water level fluctuations mostly occur during the night time. Nevertheless, Fig. 15 shows our model is able to reliably capture them to build the flood hydrograph.

Table 2 summarizes our estimation errors in field experiments 2∼4 by comparing our estimations and the groundtruth depths collected from the nearest gages. The estimation errors mainly come from two factors: (1) the accuracy of water and reference object segmentation, and (2) the accuracy of reference template modeling and matching. For example, in the Boston Harbor scenes, the reference object (building) and water region are very far away from the camera. A single pixel error in both water segmentation and reference object localization could lead to a relatively bigger estimation error. Therefore, in general, our estimation performs better when the camera is close to the flood scene and reference objects. Severe weather or bad illumination conditions poses extra challenges. For example, during Hurricane Harvey, water level fluctuations mostly occur in the nighttime. The water was hardly

Table 2
Water depth/level estimations from the testing videos of the collected WaterDataset. Errors E_A are measured by the absolute difference between the estimated and groundtruth values in centimeters. The relative errors E_R are computed by normalizing the absolute errors, i.e., E divided by the maximum groundtruth water depth.

Location	Date	E_A (cm)	E_R (%)
LSU Creek	2020/05/26	2.3	2.3
LSU Creek	2020/06/24	1.8	1.4
Boston Harbor	2019/01/19–23	31.3	7.9
Boston Harbor	2020/06/17–19	37.5	12.4
Boston Harbor	2022/01/29–31	34.4	8.7
Hou. Buffalo Bayou	2017/08/25–31	91.2	11.0

[4] STN Site No. TXHAR23198 in USGS.



(a) One frame of the input image.

(b) Segmentation and estimation.

(c) Quantitative comparisons.

Fig. 14. Field experiment on 2019 Boston Harbor. (a) shows the original image and (b) shows the water segmentation (blue mask), reference marker (green box), and the estimated water level (red line). (c) plots the estimated water level and the groundtruth.

visible and its segmentation is very challenging. This leads to bigger absolute estimation errors in the Houston Buffalo Bayou scene.

4. Discussion

4.1. Implementation details

For *image segmentation*, we use EfficientNet-B4 and LinkNet (Yang et al., 2018; Chaurasia and Culurciello, 2017) as the segmentation architecture. The model was then trained using images from our WaterDataset. All the training images were resized to the network's input resolution of 416 × 416 and fed through the network in a batch size of 4. The learning rate was initialized to 10^{-4}, and decreased to 1e–5 halfway through training. The widely used metric mean Intersection-over-Union (mIoU) (Liang et al., 2020a) was adopted to evaluate the segmentation quality. All these three segmentation networks were trained for 150 epochs using dice loss (Sudre et al., 2017).

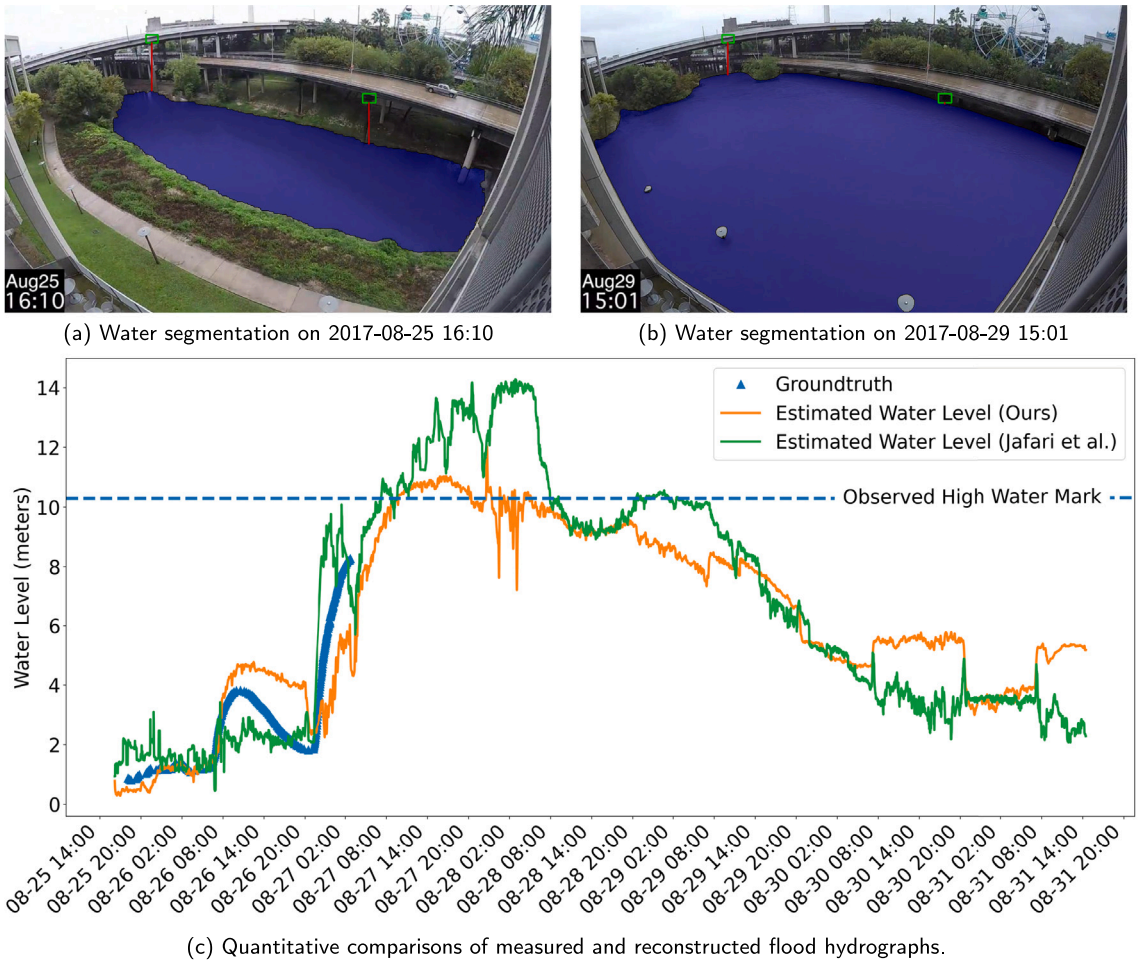
For *video segmentation*, we used AFB-URR (Liang et al., 2020b) as the architecture. It was pre-trained on the YouTube-VOS (Xu et al., 2018) dataset which contains 5K videos of the general objects. We then fine-tuned it on our WaterDataset. The initial learning rate was $1e{-}5$, and it was halved every 25 epochs. The total number of training epochs was 100. We chose the AdamW (Loshchilov and Hutter, 2017) as the back-propagation optimizer. The input resolution of the video is 480 × 854.

Note that in both image- and video-based flood segmentation, we resize the input to meet the requirement of the segmentation module. Then we resize the estimated water mask back to the original resolution. Since the resolution range of most video cameras is from 480p to 1080p. Our segmentation modules balance the speed and segmentation quality, which are insensitive to the input resolution.

For the reference object segmentation, we used the Mask-RCNN to segment the stopsign and people (He et al., 2017). We used MeshTransformer to complete the 3D mesh of the detected person (Lin et al., 2021). We used CSRT Tracker to track the user-selected bounding box in the input video (Lukezic et al., 2017). We directly used their model and pretrained weights in the experiments.

4.2. Limitations and future work

A key challenge in image/video segmentation is the processing of nighttime images/videos. When the scene becomes dark and dim, specular highlights and reflections could significantly affect water region identification and segmentation. Although with the help of spatio-temporal coherency and adaptive feature updating, our video segmentation model can identify the main water regions, their boundaries are less accurately tracked. Such less accurate contours could consequently



(a) Water segmentation on 2017-08-25 16:10

(b) Water segmentation on 2017-08-29 15:01

(c) Quantitative comparisons of measured and reconstructed flood hydrographs.

Fig. 15. Field experiment on 2017 Houston flooding. (a) Water segmentation on August 25; (b) Water segmentation on August 29, where water-(blue mask), reference marker - (green box), and flood level estimator - (red line) are shown. (c) Comparison of the estimated water levels (in orange) by V-FloodNet, the estimated water levels by the previous model (Jafari et al., 2021) (in green), and the groundtruth (in blue).

affect water level estimation. In future work, we will incorporate video-based dark image-enhancing techniques to tackle this issue.

The segmentation of the first frame affects the accuracy of the entire video segmentation. To make this first-frame segmentation more reliable, the training dataset should contain flood and water images in various environments, with different weather, illumination, and reflection conditions. our WaterDataset currently contains several thousand water images but there are only about 150 flood images. We will collect and label more flood images, especially those whose flood region is difficult to detect or segment. This will improve the accuracy and robustness of our image segmentation model.

5. Conclusions

We proposed a comprehensive video processing and analysis system that can detect and segment water and surrounding reference objects, and use them to estimate the water level or inundation depth. Unlike existing segmentation techniques in the computer vision literature that were developed to handle common daily objects, our system is capable of adaptively capturing the appearance change of water in the scene under dynamic/noisy and weather/lighting conditions. Hence, it can produce more accurate and precise water segmentation, which is crucial to flood height or inundation depth estimation. We also developed a depth estimation algorithm using reference objects and template matching, allowing the system to effectively calculate the inundation depth. This pipeline can run automatically, or if necessary, interactively, to support convenient water depth estimation and monitoring. In our experiments, we demonstrated the effectiveness and robustness of the proposed system in extensive applications. Validated through multiple field experiments, our system can provide effective water depth estimations in various video scenes. Besides the methodology, we also released a set of water-related image and video data with annotations. These data can benefit the training of other deep learning-based water analysis systems. We also made our code and models open-sourced. They can conveniently be used as enabling tools or used for comparative studies.

Code and data availability

The source code is open-source for academic research. It is available at https://github.com/xmlyqing00/V-FloodNet. Detailed installation and usage instructions are included in the README file.

The proposed WaterDataset data and pretrained model weights are also available for downloading.

CRediT authorship contribution statement

Yongqing Liang: Methodology, Software, Writing, Visualization. **Xin Li:** Methodology, Resources, Writing, Supervision, Funding acquisition. **Brian Tsai:** Software, Data Curation, Writing, Visualization. **Qin Chen:** Conceptualization, Validation, Resources, Writing, Project administration, Funding acquisition. **Navid Jafari:** Validation, Investigation, Resources, Writing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All our developed codes and data are published on GitHub for public usage, with comprehensive usage instructions. See the end of the abstract.

Acknowledgments

We would like to thank Thomas Rinaudo, Claire White, Amina Meselhe, and Dominion Ajayi who helped label the water images and videos. We thank Alex Wu who developed parts of the people mesh registration model. We thank Eli Barbin who conducted the experiments in the LSU Lake. This material is based upon work supported by the National Science Foundation, USA (Grant #1760582, #2139882, #2139883, and #1946231). The authors would like to thank Louisiana Sea Grant Undergraduate Research Opportunities Program (UROP), USA, T. Baker Smith, Inc., Louisiana Board of Regents Industrial Ties Research Program LEQSF (2018-21)-RD-B-03, and the Northeastern University Global Resilience Institute, USA for supporting this research. We appreciate permission by Mr. Teddy Vandenberg to use his time-lapse video of Buffalo Bayou during Hurricane Harvey. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, T. Baker Smith, Louisiana Board of Regents, and Louisiana Sea Grant.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.envsoft.2022.105586.

References

Aerts, J.C., Botzen, W.W., Emanuel, K., Lin, N., De Moel, H., Michel-Kerjan, E.O., 2014. Evaluating flood resilience strategies for coastal megacities. Science 344 (6183), 473–475.

Alizadeh, B., Li, D., Zhang, Z., Behzadan, A.H., 2021. Feasibility study of urban flood mapping using traffic signs for route optimization. arXiv:2109.11712 [Cs], URL: http://arxiv.org/abs/2109.11712. arXiv: 2109.11712.

Bao, L., Wu, B., Liu, W., 2018. CNN in MRF: Video object segmentation via inference in a CNN-based higher-order spatio-temporal MRF. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, Salt Lake City, UT, pp. 5977–5986. http://dx.doi.org/10.1109/CVPR.2018.00626, URL: https://ieeexplore.ieee.org/document/8578724/.

Chaudhary, P., D'Aronco, S., Moy de Vitry, M., Leitão, J.P., Wegner, J.D., 2019. Flood-water level estimation from social media images. In: ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. IV, Copernicus Publications, pp. 5–12. http://dx.doi.org/10.5194/isprs-annals-IV-2-W5-5-2019, URL: https://www.research-collection.ethz.ch/handle/20.500.11850/351581.

Chaurasia, A., Culurciello, E., 2017. LinkNet: Exploiting encoder representations for efficient semantic segmentation. In: 2017 IEEE Visual Communications and Image Processing (VCIP). pp. 1–4. http://dx.doi.org/10.1109/VCIP.2017.8305148.

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1800–1807. http://dx.doi.org/10.1109/CVPR.2017.195.

Colgan, C.S., 2017. Financing natural infrastructure for coastal flood damage reduction.

Erfani, S.M.H., Wu, Z., Wu, X., Wang, S., Goharian, E., 2022. ATLANTIS: A benchmark for semantic segmentation of waterbody images. Environ. Model. Softw. 149, 105333.

Ford, A., Barr, S., Dawson, R., Virgo, J., Batty, M., Hall, J., 2019. A multi-scale urban integrated assessment framework for climate change studies: A flooding application. Comput. Environ. Urban Syst. 75, 229–243. http://dx.doi.org/10.1016/j.compenvurbsys.2019.02.005, URL: https://www.sciencedirect.com/science/article/pii/S0198971518304708.

Fryar, C.D., Kruszon-Moran, D.M., Gu, Q., Ogden, C.L., 2018. Mean body weight, height, waist circumference, and body mass index among adults: United States, 1999–2000 through 2015–2016. Nat. Health Stat. Rep. 122, 1–16.

Geetha, M., Manoj, M., Sarika, A.S., Mohan, M., Rao, S.N., 2017. Detection and estimation of the extent of flood from crowd sourced images. In: 2017 International Conference on Communication and Signal Processing (ICCSP). pp. 0603–0608. http://dx.doi.org/10.1109/ICCSP.2017.8286429.

Hartley, R., Zisserman, A., 2003. Multiple View Geometry in Computer Vision. Cambridge University Press.

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2961–2969.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778.

Hosseiny, H., 2021. A deep learning model for predicting river flood depth and extent. Environ. Model. Softw. 145, 105186. http://dx.doi.org/10.1016/j.envsoft.2021.105186, URL: https://www.sciencedirect.com/science/article/pii/S1364815221002280.

Hu, Y.-T., Huang, J.-B., Schwing, A., 2017. Maskrnn: Instance level video object segmentation. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30. Curran Associates, Inc., pp. 325–334, URL: http://papers.nips.cc/paper/6636-maskrnn-instance-level-video-object-segmentation.pdf.

Hu, Y.-T., Huang, J.-B., Schwing, A.G., 2018a. Videomatch: Matching based video object segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 54–70.

Hu, P., Wang, G., Kong, X., Kuen, J., Tan, Y.-P., 2018b. Motion-guided cascaded refinement network for video object segmentation, pp. 1400–1409, URL: https://openaccess.thecvf.com/content_cvpr_2018/html/Hu_Motion-Guided_Cascaded_Refinement_CVPR_2018_paper.html.

Jafari, N.H., Li, X., Chen, Q., Le, C.-Y., Betzer, L.P., Liang, Y., 2021. Real-time water level monitoring using live cameras and computer vision techniques. Comput. Geosci. 147, 104642.

Kharazi, B.A., Behzadan, A.H., 2021. Flood depth mapping in street photos with image processing and deep neural networks. Comput. Environ. Urban Syst. 88, 101628.

Liang, Y., Jafari, N., Luo, X., Chen, Q., Cao, Y., Li, X., 2020a. WaterNet: An adaptive matching pipeline for segmenting water with volatile appearance. Comput. Vis. Media 1–14.

Liang, Y., Li, X., Jafari, N., Chen, J., 2020b. Video object segmentation with adaptive feature bank and uncertain-region refinement. Adv. Neural Inf. Process. Syst. 33, 3430–3441.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft coco: Common objects in context. In: European Conference on Computer Vision. Springer, pp. 740–755.

Lin, H., Qi, X., Jia, J., 2019. AGSS-VOS: Attention guided single-shot video object segmentation. pp. 3949–3957, URL: http://openaccess.thecvf.com/content_ICCV_2019/html/Lin_AGSS-VOS_Attention_Guided_Single-Shot_Video_Object_Segmentation_ICCV_2019_paper.html.

Lin, K., Wang, L., Liu, Z., 2021. End-to-end human pose and mesh reconstruction with transformers. In: CVPR.

Long, J., Shelhamer, E., Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3431–3440.

Lopez-Fuentes, L., Rossi, C., Skinnemoen, H., 2017. River segmentation for flood monitoring. In: 2017 IEEE International Conference on Big Data (Big Data). IEEE, pp. 3746–3749.

Loshchilov, I., Hutter, F., 2017. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

Lukezic, A., Vojir, T., Čehovin Zajc, L., Matas, J., Kristan, M., 2017. Discriminative correlation filter with channel and spatial reliability. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6309–6318.

Maninis, K.-K., Caelles, S., Chen, Y., Pont-Tuset, J., Leal-Taixe, L., Cremers, D., Van Gool, L., 2019. Video object segmentation without temporal information. IEEE Trans. Pattern Anal. Mach. Intell. 41 (6), 1515–1530. http://dx.doi.org/10.1109/TPAMI.2018.2838670, Place: United States Publisher: IEEE Computer Society.

Mao, X., Chow, J.K., Su, Z., Wang, Y.-H., Li, J., Wu, T., Li, T., 2021. Deep learning-enhanced extraction of drainage networks from digital elevation models. Environ. Model. Softw. 144, 105135. http://dx.doi.org/10.1016/j.envsoft.2021.105135, URL: https://www.sciencedirect.com/science/article/pii/S136481522100178X.

Meng, Z., Peng, B., Huang, Q., 2019. Flood depth estimation from web images. In: Proceedings of the 2nd ACM SIGSPATIAL International Workshop on Advances on Resilient and Intelligent Cities. pp. 37–40.

Ning, H., Li, Z., Hodgson, M.E., Wang, C.S., 2020. Prototyping a social media flooding photo screening system based on deep learning. ISPRS Int. J. Geo-Inf. 9 (2), http://dx.doi.org/10.3390/ijgi9020104, URL: https://www.mdpi.com/2220-9964/9/2/104.

Oh, S.W., Lee, J.-Y., Sunkavalli, K., Kim, S.J., 2018. Fast video object segmentation by reference-guided mask propagation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7376–7385. http://dx.doi.org/10.1109/CVPR.2018.00770, ISSN: 2575-7075.

Oh, S.W., Lee, J.-Y., Xu, N., Kim, S.J., 2019. Video object segmentation using space-time memory networks. pp. 9226–9235, URL: http://openaccess.thecvf.com/content_ICCV_2019/html/Oh_Video_Object_Segmentation_Using_Space-Time_Memory_Networks_ICCV_2019_paper.html.

Pally, R., Samadi, S., 2022. Application of image processing and convolutional neural networks for flood image classification and semantic segmentation. Environ. Model. Softw. 148, 105285.

Park, S., Baek, F., Sohn, J., Kim, H., 2021. Computer vision–based estimation of flood depth in flooded-vehicle images. J. Comput. Civ. Eng. 35 (2), 04020072.

Perazzi, F., Khoreva, A., Benenson, R., Schiele, B., Sorkine-Hornung, A., 2017. Learning video object segmentation from static images. In: Computer Vision and Pattern Recognition.

Sudre, C.H., Li, W., Vercauteren, T., Ourselin, S., Jorge Cardoso, M., 2017. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support. Springer, pp. 240–248.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–9. http://dx.doi.org/10.1109/CVPR.2015.7298594.

Tan, M., Le, Q., 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (Eds.), Proceedings of the 36th International Conference on Machine Learning, Vol. 97. PMLR, pp. 6105–6114, URL: https://proceedings.mlr.press/v97/tan19a.html.

U.S. Department of Transportation, Federal Highway Administration, 2003. Manual on uniform traffic default controls (MUTCD) - 2003 edition revision 1 chapter 2B. URL: https://mutcd.fhwa.dot.gov/htm/2003r1/part2/part2b1.htm.

U.S. Geological Survey, 2017. USGS flood event viewer - short-term network data portal. URL: http://water.usgs.gov/floods/FEV/.

Voigtlaender, P., Chai, Y., Schroff, F., Adam, H., Leibe, B., Chen, L.-C., 2019. Feelvos: Fast end-to-end embedding learning for video object segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9481–9490.

Wada, K., 2021. Labelme: Image polygonal annotation with python, http://dx.doi.org/10.5281/zenodo.5711226, URL: https://github.com/wkentaro/labelme.

Wang, Z., Xu, J., Liu, L., Zhu, F., Shao, L., 2019. RANet: Ranking attention network for fast video object segmentation. pp. 3978–3987, URL: http://openaccess.thecvf.com/content_ICCV_2019/html/Wang_RANet_Ranking_Attention_Network_for_Fast_Video_Object_Segmentation_ICCV_2019_paper.html.

Xu, Z., Wang, S., Stanislawski, L.V., Jiang, Z., Jaroenchai, N., Sainju, A.M., Shavers, E., Usery, E.L., Chen, L., Li, Z., Su, B., 2021. An attention U-net model for detection of fine-scale hydrologic streamlines. Environ. Model. Softw. 140, 104992. http://dx.doi.org/10.1016/j.envsoft.2021.104992, URL: https://www.sciencedirect.com/science/article/pii/S1364815221000359.

Xu, N., Yang, L., Fan, Y., Yang, J., Yue, D., Liang, Y., Price, B., Cohen, S., Huang, T., 2018. Youtube-vos: Sequence-to-sequence video object segmentation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 585–601.

Yang, L., Wang, Y., Xiong, X., Yang, J., Katsaggelos, A.K., 2018. Efficient video object segmentation via network modulation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. IEEE, Salt Lake City, UT, pp. 6499–6507. http://dx.doi.org/10.1109/CVPR.2018.00680, URL: https://ieeexplore.ieee.org/document/8578778/.

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A., 2017. Scene parsing through ade20k dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 633–641.