



Segmentation and deep learning to digitalize the kinematics of flow-type landslides

Clarence E. Choi¹ · Zhengyu Liang¹

Received: 19 May 2023 / Accepted: 20 December 2023 / Published online: 2 April 2024
© The Author(s) 2024

Abstract

Flow-type landslides, including subaerial and submarine debris flows, have poor spatiotemporal predictability. Therefore, researchers rely heavily on experimental evidence in revealing complex flow mechanisms and evaluating theoretical models. To measure the velocity field of experimental flows, conventional image analysis tools for measuring soil deformation and hydraulics have been borrowed. However, these tools were not developed for capturing the kinematics of fast-moving soil–water mixtures over complex terrain under non-uniform lighting conditions. In this study, a new framework based on deep learning was used to automatically digitalize the kinematics of experimental flow-type landslides. Captured images were broken into sequences and binarized using a fully convolutional neural network (FCNN). The proposed framework was demonstrated to outperform classic image processing algorithms (e.g., particle image velocimetry, trainable Weka segmentation, and thresholding algorithms) over a wide range of experimental conditions. The FCNN model was even able to process images from consumer-grade cameras under complex shadow, light, and boundary conditions. This feature is most useful for field-scale experimentation. With fewer than 15 annotated training images, the FCNN digitalized experimental flows with an accuracy of 97% in semantic segmentation.

Keywords Artificial intelligence · Granular materials · Landslides · Soil/structure interaction

1 Introduction

Flow-type landslides, such as debris flows, are rapidly moving particle–fluid mixtures that pose a threat to sustainable development. These landslides have poor spatiotemporal predictability, such that experiments [6] and numerical simulations [18] are often conducted to reveal underlying flow mechanisms and evaluate theoretical models. Flume experiments are a commonly employed source of physical evidence that can be utilized by numerical modelers to calibrate their numerical simulations. For instance, de Haas et al. [9] conducted laboratory flume experiments to investigate the runout and deposition behaviors of debris flow mixtures. Iverson et al. [15] analyzed the flow volume and flow runout distance using a 120-m-long flume. Debris flows impacting barriers were

also investigated in field-scale experiments [23, 33] using a 28-m-long field-scale flume. The model flows in these studies were highly transient. For example, flows in field-scale experiments [23, 33] required less than 5 s to travel from the storage container to the outlet of the channel. Owing to the high flow velocities, the window for obtaining measurements of the flow kinematics (e.g., the frontal velocity) is limited. Therefore, high-speed cameras that capture images at high frame rates are needed for image analysis. For example, Du et al. [10] use 300 frames/s and Chen et al. [5] use 240 frames/s in their experiments. However, the use of high-speed cameras is often limited to controlled laboratory environments with consistent lighting conditions. It is difficult and time-consuming to deploy high-speed cameras in field-scale experiments. To address this challenge, researchers have started using consumer-grade cameras as an auxiliary tool in both lab-scale and field-scale experiments to extract information from experimental flows while reducing the complexity and cost of data acquisition [9, 18, 23].

✉ Zhengyu Liang
lzyhku@connect.hku.hk

¹ Department of Civil Engineering, The University of Hong Kong, Pok Fu Lam, Hong Kong SAR, China

Flow videos from both high-speed and consumer-grade cameras need to be analyzed by image processing. Among image processing methods, particle image velocimetry (PIV) [35] has been extensively used to deduce the displacement field of flow materials [30]. In PIV analysis, video frames are divided into a grid of patches. The displacement field is then calculated by comparing the texture of patches between two frames. The texture processed by PIV can be either the texture of materials or artificial textures (e.g., textures obtained by adding tracers). Another method often compared with PIV is particle tracking velocimetry (PTV) [22]. Instead of comparing textures of patches, PTV tracks the velocity of tracer particles. Except for instances where sufficiently large grains are present and observable, as reported in [30, 31], PTV requires the addition of tracer particles with sufficient color contrast against the color of the flow materials. The number of tracer particles to add is not easy to establish. On the one hand, enough tracers should be added so that they can be uniformly distributed in the flow and observed in images. On the other hand, the number of tracers should be limited so that the properties of the flow (e.g., the solid fraction or bulk density) do not change. Compared with PTV, PIV usually does not require additional tracer particles for granular flows [30]. However, tracers must be added if the experimental materials being examined by PIV do not have distinguishable textures (e.g., clay) [10]. For methods like PIV or PTV, which track textures or tracers, tracking algorithms are readily affected by the ambient light conditions and shadows. Consequently, costly light-emitting diode lighting devices without flicker are needed, and ambient light conditions need to be well controlled. However, light conditions are difficult to control in field-scale experiments [23, 33].

With the rapid development of artificial intelligence, object detection techniques based on deep learning have been increasingly used in recent years for tracking the bounding box of flow boundaries. For example, eight types of convolutional neural network (CNN) algorithms were used to detect flooded areas in aerial images [25]. The object detection neural network YOLOv4 was used to detect flow bounding boxes from the images of real geohazards and laboratory-scale flume experiments in [24]. Bounding boxes are an important advancement in flow tracking. However, bounding boxes often fail to detect the boundaries of flows with complex frontal morphologies (i.e., unchanneled flow fronts with fingers). Flow boundary tracking is often equivalent to the machine vision process of segmenting images into backgrounds and flow bodies. Segmentation is a classification task based on digital image processing whereby each pixel of an image is classified as either a background pixel or a flow pixel. One popular method that is used to classify pixels is thresholding, which

distinguishes a flow and its background according to color-related properties [21]. However, simple thresholding methods struggle to process complex images. Specifically, recent observations [18] suggested that classical thresholding typically yields noisy results when applied to an image with a complex background with shadows and reflected light. In comparison, a deep learning-empowered image segmentation method may outperform classical thresholding and yield a clean and accurate result.

The objective of this study is to create a workflow that assists physical modelers in effectively generating digital representations of flow-type landslides in experiments. The workflow will allow for the maximum amount of information to be extracted from flow videos of these time-consuming experiments. To accomplish this objective, a new framework, empowered by a deep learning algorithm, that handles complex lighting conditions encountered in both laboratory-scale and field-scale experiments is introduced. The framework was designed to be easy-to-use and can operate with consumer-grade cameras. To evaluate the proposed framework, three experiments, namely a field-scale flume experiment, a submarine experiment, and an unchanneled experiment, were analyzed. These experiments generated experimental flow videos under different flow and light conditions.

2 Segmentation of flows using a deep FCNN

2.1 Pre-processing of images

The framework adopted in this study uses video captured by either research- or consumer-grade cameras as input. The recorded video is separated into image sequences and then processed, and the flow kinematics are computed. To ensure that the computed flow kinematics are accurate, images of the flow should be free from distortion. However, cameras produce images with perspective distortion such that there is a fisheye effect. The fisheye effect is a byproduct of the lens and increases the field of view (FOV). In this study, an algebraic algorithm [1] was used to perform fisheye correction. Furthermore, the perspective was corrected to treat perspective distortion (i.e., objects farther from the lens being larger than those that are closer). As the prerequisite of perspective correction, an affine transformation matrix C is first calculated:

$$\begin{pmatrix} wx'_i \\ wy'_i \\ w \end{pmatrix} = C \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}, \quad (1)$$

where the vector $(wx'_i \ wy'_i \ w)^T$ gives the homogenous coordinates of the i th pixel after transformation and

$(x_i \ y_i \ 1)^T$ gives the homogenous coordinates of the i th pixel before the transformation. The homogenous coordinates $(x_i \ y_i \ 1)^T$ are constructed from the vector of coordinates of a pixel $(x_i \ y_i)^T$ by appending an extra dimension with unity magnitude to the vector. After transformation, the extra dimension is transformed into a random number w . The value of w depends on the transformation matrix C , which corrects the errors induced by camera rotation and translation. C is given as follows:

$$\begin{bmatrix} R_1 & R_2 & T_1 \\ C_1 & C_2 & T_2 \\ W_1 & W_2 & 1 \end{bmatrix}, \quad (2)$$

where the first column $(R_1 \ C_1 \ W_1)^T$ corrects the vertical distortion, the second column $(R_2 \ C_2 \ W_2)^T$ corrects the horizontal distortion, T_1 corrects distortion due to vertical camera translation, and T_2 corrects distortion due to horizontal camera translation. To obtain the parameters in Eq. 2, a set of linear equations [36] is solved:

$$\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1y'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2y'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3y'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4y'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4 \end{bmatrix} \times \begin{bmatrix} R_1 \\ R_2 \\ T_1 \\ C_1 \\ C_2 \\ T_2 \\ W_1 \\ W_2 \end{bmatrix}, \quad (3)$$

where x_1, x_2, x_3, x_4 and y_1, y_2, y_3, y_4 are the coordinates of four user-selected reference points on a distorted image and x'_1, x'_2, x'_3, x'_4 and y'_1, y'_2, y'_3, y'_4 are the corresponding corrected coordinates of the user-selected reference points. The perspective distortion of an image is reduced by applying perspective correction to each pixel.

2.2 Segmentation of flows using a deep FCNN

For experiments where the entire flow body needs to be measured, flow boundaries need to be identified from each frame of the recorded video after image correction (as

described in Sect. 2.1). Here, semantic segmentation is first adopted to separate corrected images into flow (i.e., white) and background (i.e., black). The boundary outlining a flow is then identified using the Moore neighborhood tracing algorithm [12]. To perform semantic segmentation, an FCNN model modified from U-net [27] is used. This modified U-net boasts a reduced parameter count when compared to its original version, making the modified model a more lightweight alternative to the original U-net. According to [11, 19], the modification results in a less computationally demanding model without compromising the accuracy of the model segmentation capabilities. Figure 1 shows the workflow of FCNN semantic segmentation using a sample image of clay–water mixture flowing through a model forest (as discussed in Sect. 3). First, the raw flow video frame, which is distorted, is corrected before being used as the input of the FCNN model. However, since the FCNN model only accepts images of a fixed size (i.e., $720 \times 720 \times 3$ pixels), flow images of arbitrary sizes cannot be directly input into the FCNN. To address this issue, a sliding window strategy is adopted to scan the full-size flow image (2704×2028 pixels) following a zigzag trajectory indicated by the green arrows. At each position, the window crops the image, which is then passed through the FCNN encoding structures to extract the feature maps. The feature maps are then used to differentiate between the flow and background, and subsequently decoded. The encoding and decoding part of the FCNN have similar structures except that the decoding parts use the up-sampling operation as the last operation instead of max-pooling [7]. Detailed architectures of the encoding and decoding blocks parts are discussed in Appendix A. The decoding blocks output the semantic segmentation results of the cropped image, corresponding to the current sliding window position (i.e., the position of the green rectangle). To process a full-size image, encoding and decoding are repeatedly conducted at all positions on which the sliding window lands.

2.3 Training

The FCNN needs to be trained before semantic segmentation. Here, to train the network, flow images were randomly chosen from frames of a video of an experiment as training data. These training images were annotated using an open-source annotation tool LabelMe [28]. The flow on the images was presented in white and the background in black. This black and white presentation was used as the ground truth for semantic segmentation, which separated the images into the flow and background. The images were then cropped into sub-images using the sliding window. The FCNN only uses images of fixed size ($720 \times 720 \times 3$ pixels) as input, and the cropped images must be of a size

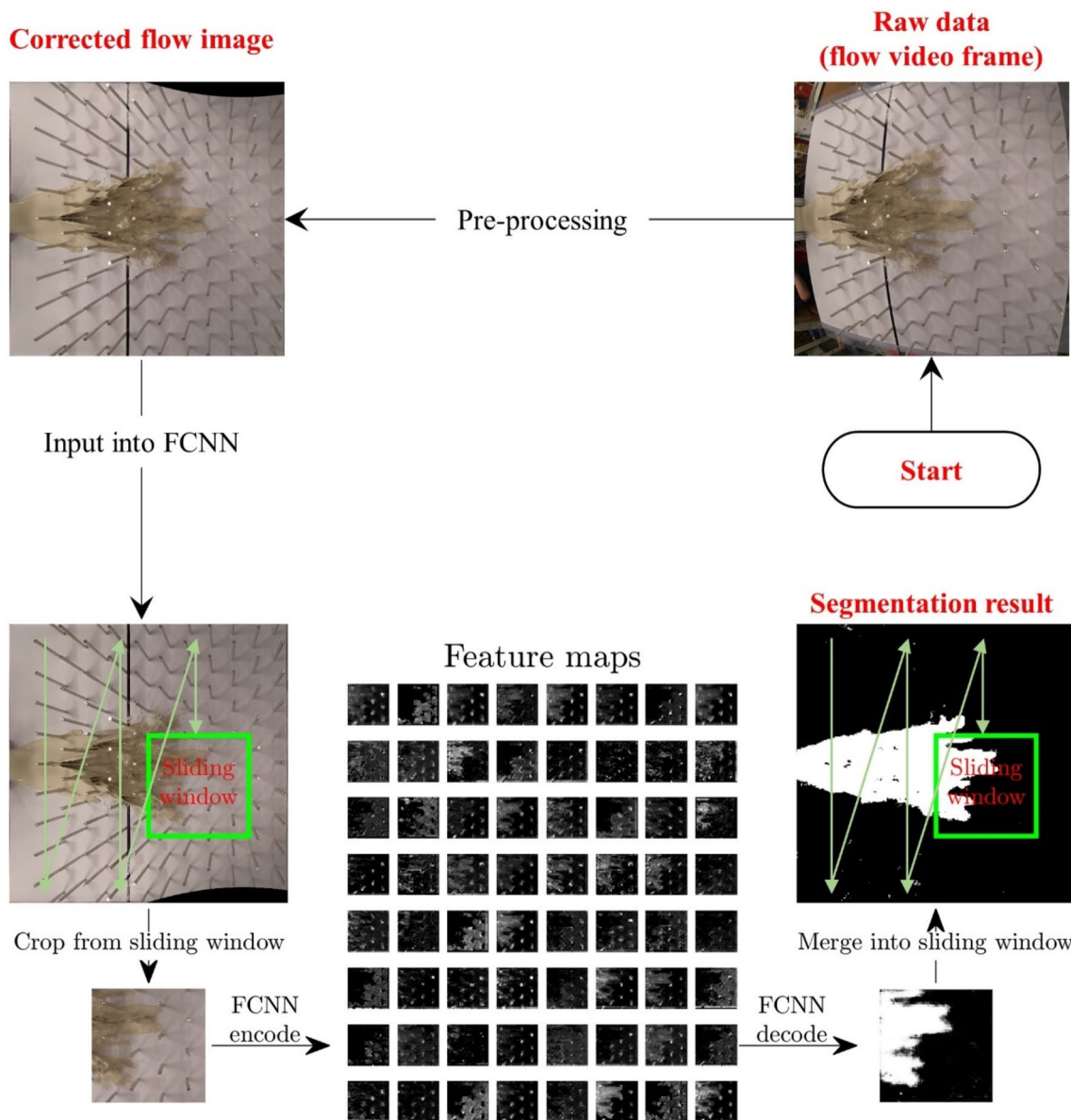


Fig. 1 Example of semantic segmentation using the fully convolutional neural network (FCNN), where the input image is photographed in the unchanneled experiment of clay–water flow through a model forest (as discussed in Sect. 3)

that fits the network input. Following the literature [19], 90% of the cropped images were fed into an Adam optimizer [16] to train the network. These images are called training sets. The remaining 10% of images (i.e., the validation set) were used for evaluating the accuracy of the segmentation during training. To enhance the ability of the network to process images with noise, data augmentation [37] was performed on the training images by adding random noise. Finally, training images cropped from randomly selected frames of flow video was combined with the black and white presentations of the images. The training was performed on an RTX 3090 graphical processing unit for up to 100 epochs. During training, the

learning rate was set at 1.000×10^{-5} and the batch size was set at 10, and the binary cross-entropy [7] function was used as the loss function.

3 Experimental evidence for evaluating FCNN

3.1 Field-scale channelized debris flow

Images recorded in three different experiments were used to evaluate the proposed FCNN model. Details of each experiment are discussed below. The first experiment was

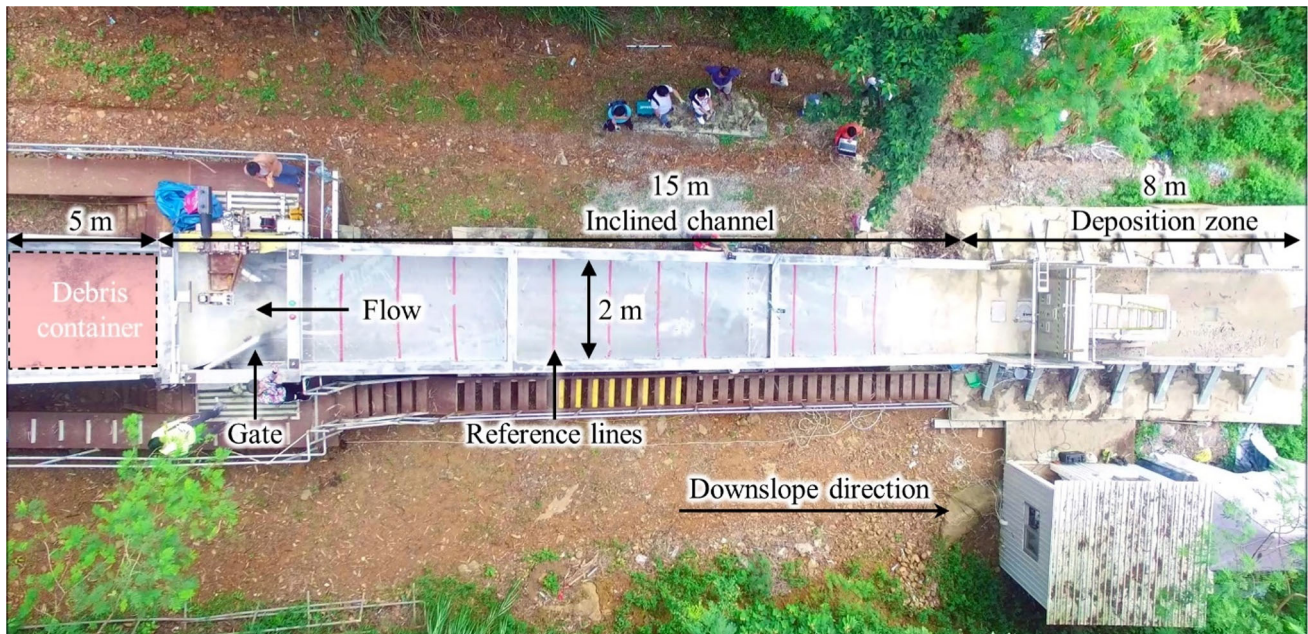


Fig. 2 Aerial photograph of the 28-m-long flume facility at the Kadoorie Centre of the University of Hong Kong [23]

performed in [23] using a field-scale flume. Figure 2 shows the flume, which was 28 m in length and 2 m in width. The upstream section was a 5-m-long debris container with a slope of 30° . The container was used to store a total volume of 2.5 m^3 of debris material comprising gravel, sand, clay, and water with volumetric concentrations of 21, 36, 2, and 41%, respectively behind double gates. Immediately downstream of the gate was a channel inclined at 20° with reference lines were painted at intervals of 1 m on the bed to track the movement of the flow front in the video. At the downstream end of the flume was a deposition zone. An unmanned aerial vehicle (UAV) with an onboard consumer-grade camera was used to capture the entire flow body as it moved downslope. The captured video had a frame rate of 30 frames per second and a resolution of 1920×1080 pixels.

3.2 Submarine debris flow

The experiment of [10] modeled submarine debris flow using a laboratory-scale flume. Figure 3 shows a side view of the flume, which comprised a watertight tank with transparent side walls. The tank was 3 m in length, 0.2 m in width, and up to 1 m in height. A container that stored up to 0.08 m^3 of debris was installed at the upstream end of the model. The debris was retained by a pneumatically controlled gate. Immediately downstream of the gate was a channel with two sections, one inclined at 18° and the other inclined at 3° . A high-speed camera was mounted at the side of the 3° incline to record the flow kinematics. The camera recorded images with a resolution of 1696×642

pixels at a frame rate of 300 fps. The debris materials used were mixtures of water, Toyoura sand, and kaolinite clay. The Toyoura sand contained less than 2% of dyed particles by volume. In the experiment [10], dyed particles were used as tracers for performing PIV analysis. Table 1 summarizes the test conditions reported in [10] for evaluating the FCNN model. Two specific test conditions were selected, involving flows with varying volumetric content (vol%) of kaolinite clay, sand, and water. These variations in composition result in different flow appearances, which can be used to assess the generalizability of FCNN in flow segmentation.

3.3 Laboratory-scale model forest impacted by slurry

The bulk of laboratory-scale models of flow-type landslides have confined channels. However, for some problems (e.g., flow spreading in forests [3, 21]), unchanneled experiments are carried out. Unchanneled experiments allow lateral spreading and thus greater spatiotemporal variability in the flow height and velocity along the lateral direction than channeled experiments. Moreover, in contrast with the case for channeled flows, a larger FOV is required to capture the spreading behavior of unchanneled flows. To evaluate the FCNN model in terms of capturing the frontal kinematics of unchanneled flows, experimental cases of slurry flow impacting a model forest were analyzed in the present study. Figure 4 shows an oblique view of a model forest, which comprised an array of aluminum cylinders that were 150 mm in height and 20 mm in diameter. The cylinders

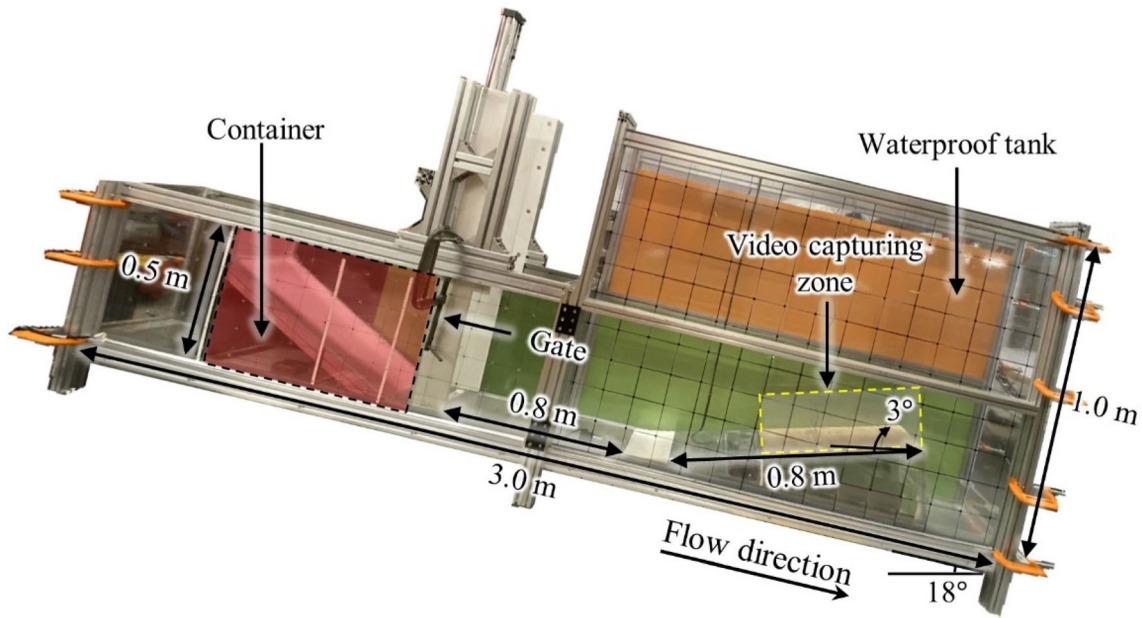


Fig. 3 Side view of the experimental setup for submarine debris flow [10]

Table 1 Reported test conditions [20] used to evaluate the FCNN model in the present study

Clay content (vol%)	Sand content (vol%)	Water content (vol%)	Bulk density (kg/m ³)
4	36	60	1,685
12	28	60	1,601

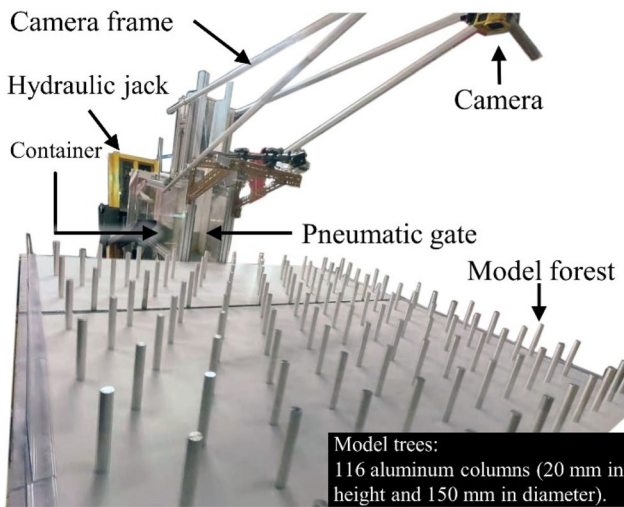


Fig. 4 Oblique-view photograph of the model forest [18]

were affixed to a 2 m × 2 m (length × width) wooden board. A hydraulic jack was used to incline the model forest. Figure 5 shows the plan view of the model forest

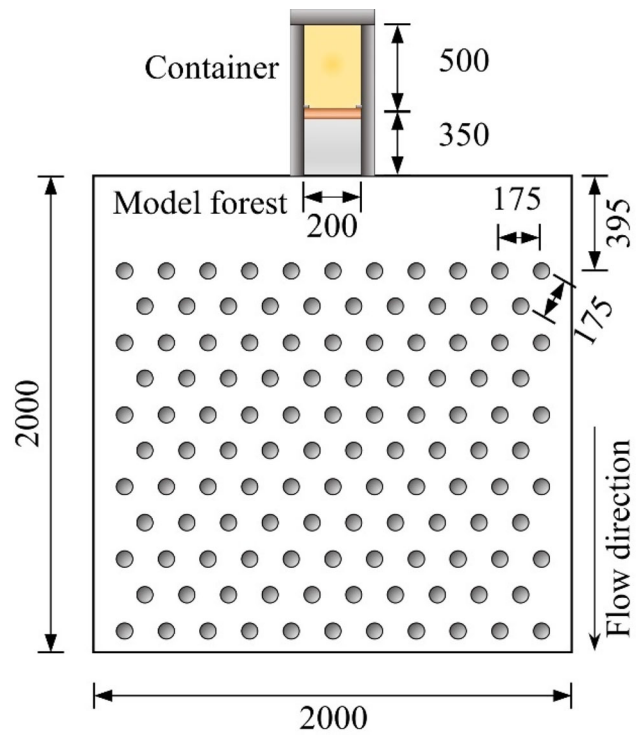


Fig. 5 Top-view schematic of the model forest with a stem spacing of 175 mm (with all dimensions in millimeters)

with a stem spacing of 175 mm. A clear-board model [13] was used for comparison in this study. Upstream of the model forest, debris material was stored in an acrylic container and released via a dam break. The debris material was retained using a pneumatically controlled gate. A

consumer-grade camera (GoPro Hero 7) was mounted above the model forest to capture the flow kinematics. The camera captured images at 50 fps and a resolution of 2704×2028 pixels.

The slurry comprised 22% kaolinite clay and 22% water volumetrically. In each experimental case, 20 kg of the mixture was prepared and placed into the storage container. To initiate a test, the pneumatically controlled gate was lifted. The retained soil–water slurry then flowed down-slope and impacted the model forest. The kinematics were captured by the camera. Two basal slopes (20° and 25°) were used in both the clear-board and model-forest experimental cases (Fig. 5).

4 Evaluation of the FCNN model

4.1 Evaluation of image correction

The unchanneled experiment was used to evaluate the image correction technique adopted in the proposed framework. A model forest was used to evaluate image correction because a panoramic view was adopted for the camera to generate an FOV large enough to capture the entire flow body. Consequently, the fisheye effect was large, making the images ideal for assessing the image correction technique adopted. Figure 6a shows a typical image of the flow with fisheye distortion. The straight edges of the wooden board (red dashed lines) were curved, such that the curvature could be used to correct for distortion. Figure 6b shows the image after fisheye correction. Although straight edges were observed, the image still suffered from perspective distortion. Specifically, the orientation of the camera was not perfectly perpendicular to the board, and the side edges of the board were thus not parallel. Such an issue can be solved by perspective correction. Perspective correction was performed by substituting four user-selected reference points (indicated by yellow diamond markers) into Eq. 3. Figure 6c shows the image after perspective correction. This image was used to perform semantic segmentation. Edges that should be parallel in reality were now parallel in the image. Moreover, the lines formed by the top of the model trees, which were parallel in reality, were now also parallel in the image.

The corrected flow image was processed using the FCNN model for semantic segmentation based on the workflow described in Fig. 1b. The flow in the image was then identified. The boundary and area of the flow were computed indirectly by counting the number of pixels belonging to the flow part of the image. The computed flow boundary and area had a unit of pixels. To transform the

unit from pixels to mm, the size s_p of a single flow pixel is calculated as follows:

$$s_p = \frac{S}{N_p}, \quad (4)$$

where S is the real size of a user-selected reference object on the image (in mm) and N_p is the number of pixels constituting this object. Taking the unchanneled experiment as an example, it was assumed that S was the width of the board, which was 2000 mm and designed to be the same as the width of the image used for semantic segmentation. As the image had a width of 2000 pixels ($N_p = 2000$), it followed that $s_p = 1$ mm/pixel.

4.2 Evaluation of the deep convolutional neural network

To evaluate the ability of the FCNN model to perform semantic segmentation, it was trained using experimental images captured in the three experiments. The experiments differed greatly in flow material and experimental setup, so training was performed separately. Table 2 summarizes the number of images used for training in each case. Table 3 summarizes the hyperparameters used to train all three experiments.

Figure 7 shows the training curves for each type of experiment. For each experiment analyzed in this study, the first 10 training epochs were the most effective. In these 10 epochs, the training accuracy increased to more than 80%. The training accuracy is defined as the percentage of correctly identified pixels, evaluated on the training set. In the cases of the field-scale and unchanneled experiments, the training accuracy increased to approximately 97% after 50 epochs of training. The validation accuracy had the same trend as the training accuracy. For the submarine experiment, the training accuracy increased to approximately 97% after 100 epochs of training. The training accuracy serves as an indicator of the chosen hyperparameters' effectiveness across all experiments. Nevertheless, the submarine debris flow experiment necessitated a greater number of training iterations due to the utilization of a high-speed camera that solely produced grayscale images. In contrast to the true-color images obtained in the field-scale and unchanneled experiments, these grayscale images provided less information for segmentation purposes. Despite this limitation, the final training accuracy of 97% demonstrates the well-trained FCNN's proficiency in performing semantic segmentations on grayscale images as well. It is worth noting that the validation accuracy for the submarine experiment is higher than the training accuracy. This interesting observation can be attributed to the presence of noise intentionally added to the training images during data augmentation, which makes it more

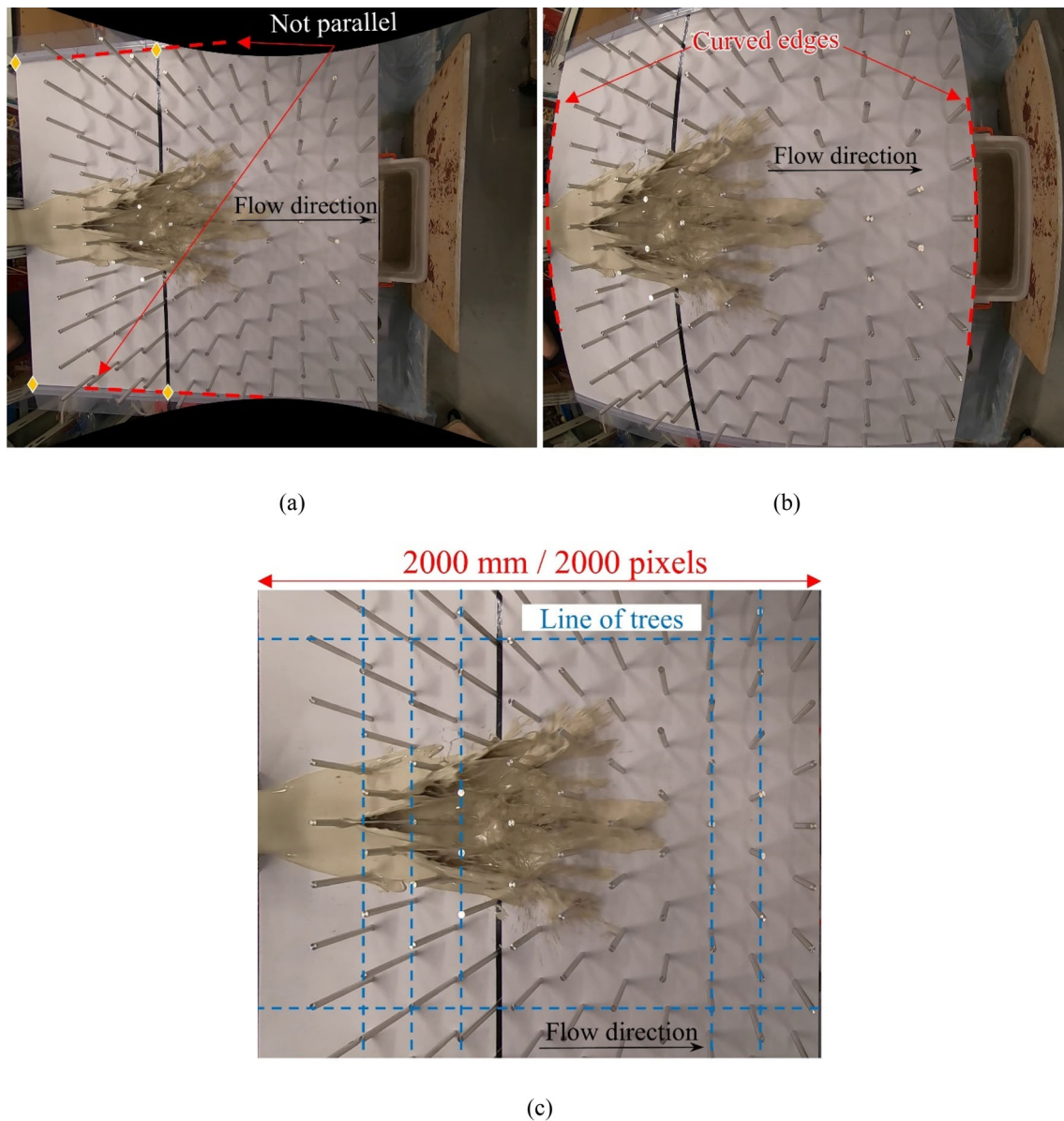


Fig. 6 Image distortion and its correction: (a) original flow image with fisheye distortion, (b) perspective distortion that distorts parallel edges, and (c) corrected image with perspective distortion

Table 2 Number of images used in training

	Annotated images (full size)	Cropped images	Cropped images used in training	Cropped images used in validation
Field-scale experiment	8	1152	1036	116
Submarine experiment	5	560	504	56
Unchanneled experiment	15	1080	972	108

Table 3 Hyperparameters used in training

Name	Value
Learning rate	$1.000 * 10^{-5}$
Adam β_1	0.900
Adam β_2	0.999
Batch size	10
Loss function	Binary crossentropy [7]

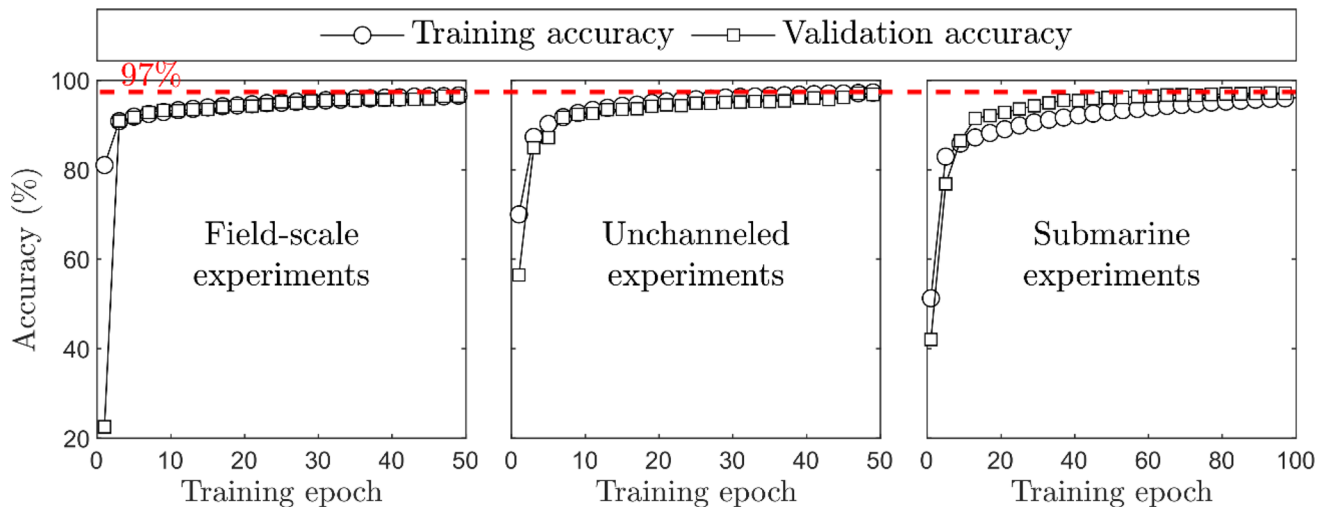


Fig. 7 Training curve of the FCNN model. The round markers represent the training accuracy and the square markers the validation accuracy

challenging for the FCNN to accurately segment them. However, in both the validation set and real experiments, this noise does not exist, resulting in higher accuracy.

Figure 8 compares the actual flow boundary obtained in the field-scale experiment and that identified using the FCNN. The boundaries of the flows (i.e., the black solid lines in Fig. 8a, b, and c and the white solid lines in Fig. 8d and e) were extracted by applying the Moore neighborhood tracing algorithm [12] to the semantic segmentation results. In the field-scale experiment (Fig. 8a), the images were taken by a consumer-grade UAV camera, which had a resolution of 1920×1080 pixels. However, the flow only occupied a small part of the image (i.e., the resolution of the area corresponding to the flow in the image was low and only 1000×165 pixels). Nevertheless, the FCNN model still identified the flow boundaries. After the flow was released (at 0.33 s), it traveled downslope with a slightly convex flow head. Specifically, the flow near the center line of the channel moved more rapidly than that near the side walls. As the flow head moved downslope, the right side of the flow traveled more rapidly than the left side. At 2.27 s, the distance traveled by the flow head on the left side was 8.2% less than that traveled by the flow head on the right side. This difference across the width of the channel was also captured by the FCNN model.

Figure 8b and c further demonstrate the ability of the FCNN model to identify the flow boundaries in the unchanneled experiment. Figure 8b shows the identified flow boundary on a laboratory-scale wooded board. Unlike the case for channeled flow, where the flow boundary was close to being rectangular, an elliptical flow boundary was observed owing to lateral spreading.

Figure 8c shows that the spatial variation in the flow velocity was even more pronounced when flow impacted the model forest. After the flow impacted a model tree, the

flow front divided into branches. In this case, accurate boundary detection was evident.

Figure 8d and e showcase the ability of the FCNN model to identify flow boundaries in the submarine experiment. In contrast with the field-scale and unchanneled experiments, a research-grade high-speed camera was adopted in the submarine experiment to capture the flow kinematics. The camera only generated grayscale images, each of which was essentially a $1696 \times 642 \times 1$ matrix. The size of the third dimension (i.e., 1) indicates that each pixel contained one parameter that described its grayscale. In comparison, the true-color images in Fig. 8a, b, and c had three parameters per pixel that described the red, green, and blue components, which constitute the color of a pixel. As such, grayscale images make it more challenging for the FCNN to identify the flow boundaries between the flowing and non-flowing parts. Notwithstanding, the FCNN model in this study output flow boundaries that were consistent with the observed boundaries. When analyzing sidewall images from landslide flume tests, it is crucial to consider the 3D effects, especially for submarine debris flows in a flume. This is because the flow front often exhibits pillowy turbulence caused by the clay content that diffuses into the water and forms a diluted layer. It is important to note that in any image-based method, including the FCNN, the captured flow edges depict the edges of the maximum projected flow contour, rather than the actual 3D flow edge at the sidewall. Hence, image pre-processing is necessary to ensure that the captured edges are accurate, as discussed in Sect. 3.1.

4.3 Evaluating field-scale debris flow

Compared with laboratory-scale experiments, research-grade high-speed cameras [10, 30] are more difficult

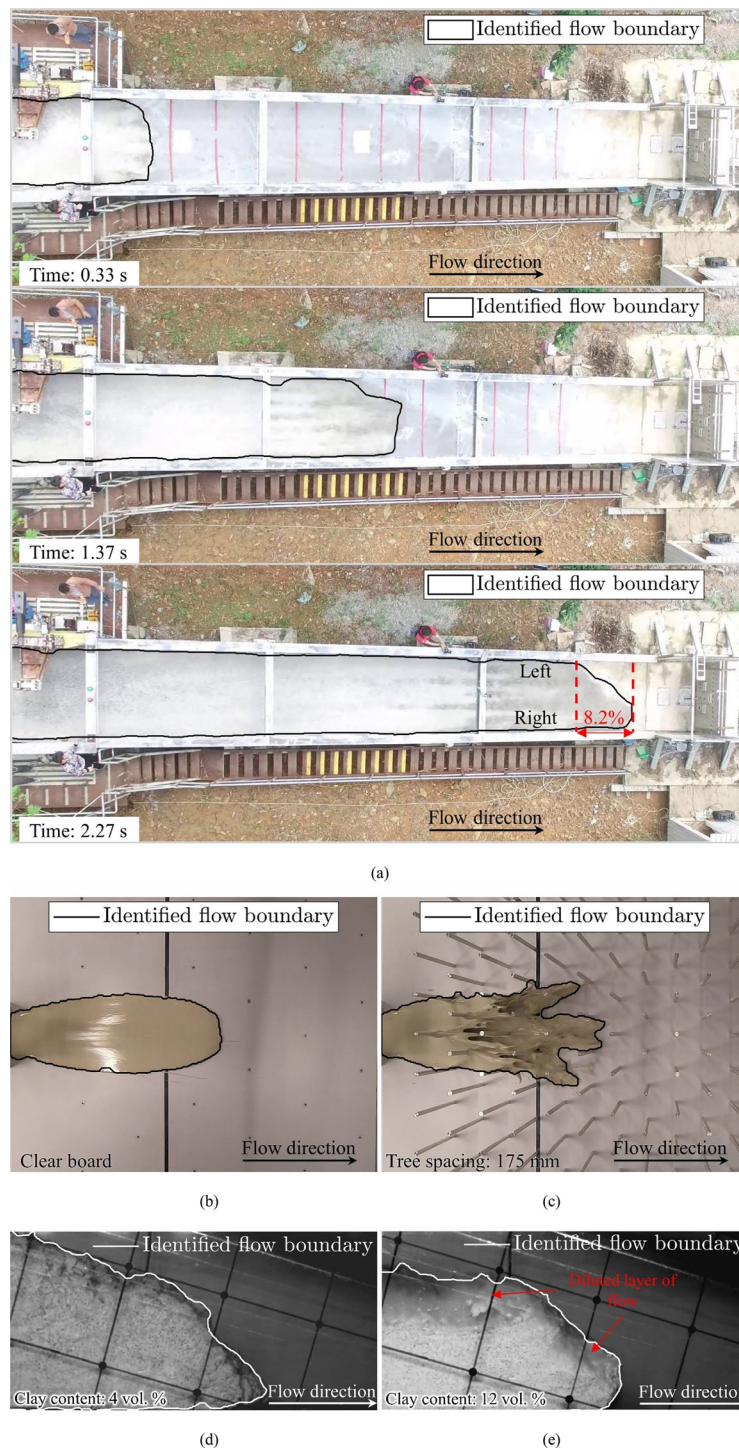


Fig. 8 Comparison of the flow boundaries identified by the FCNN model and the observed flow boundaries: **(a)** field-scale experiment, **(b)** flow spreading on a laboratory-scale clear wooden board, **(c)** flow interacting with a laboratory-scale model forest having spacing of 175 mm, **(d)** submarine flow [10] with a volumetric clay content of 4%, and **(e)** submarine flow [10] with a volumetric clay content of 12%

to deploy in the field because they require a steady power supply, real-time data transmission and processing, and well-controlled light conditions, which may not be easy to prepare in the field. Therefore,

field-scale experiments often use consumer-grade cameras [23].

Figure 9 compares the performances of different image-based measuring algorithms. Sixteen images capture from

consumer-grade cameras are illustrated. Images A to D are the raw input images. Images E to H are the semantic segmentation results produced by the FCNN using images A to D as input. Images I to L are the semantic segmentation results produced by a commonly used machine learning algorithm, the trainable Weka segmentation (TWS) algorithm [2]. TWS was implemented using ImageJ [29], which is open-source image processing software. Images M to P show the velocity field of the flow obtained through PIV [32], which was implemented using PIVlab

software [32]. Among the three algorithms, FCNN and TWS used semantic segmentation to identify the flow. However, TWS produced noisier results than the FCNN model. Discrete and isolated parts of the image that did not belong to the flow were misinterpreted. The misinterpreted parts were mixed with parts of the image that actually represented the flow, making it difficult to differentiate the flow from the background. Although the performance of TWS can be further improved by adding more training samples, the training process is laborious (refer to [2] for

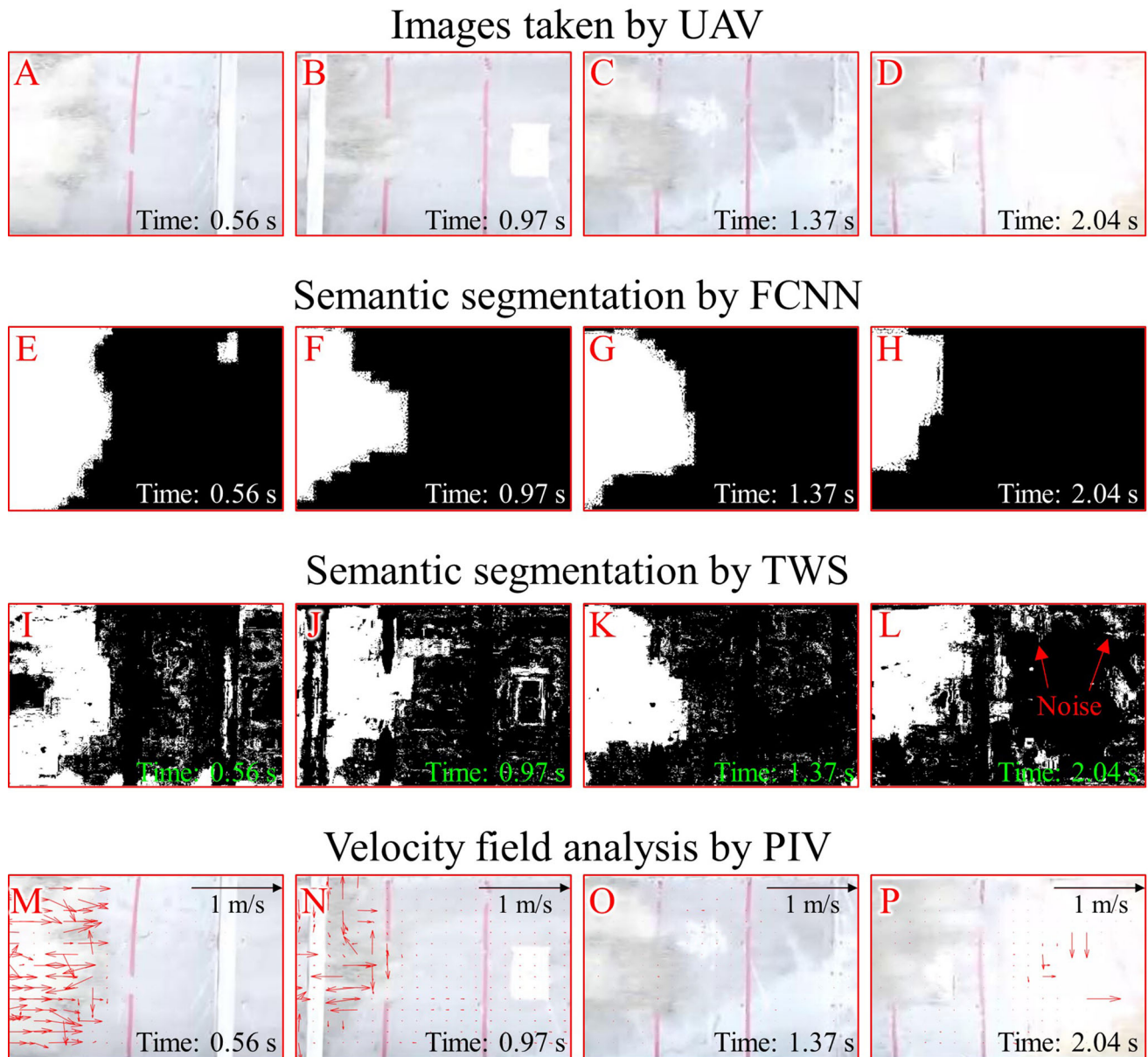


Fig. 9 Comparison of the performances of image-based measuring methods in processing images of the field-scale experiment (where the flow direction is to the right). The first row shows the source or input images of the flow head taken by an onboard camera mounted to an unmanned aerial vehicle. The second row shows the semantic segmentation results obtained using the FCNN algorithm in the present study. The third row shows the noisy results obtained by the trainable Weka segmentation (TWS) algorithm [2]. The fourth row shows the image analysis results obtained using the particle image velocimetry (PIV) algorithm [32]

the detailed training procedure). In contrast, the output of the FCNN was consistent with the real flow profile, with there being only one misinterpreted part of the flow (i.e., that at a time of 0.56 s). More importantly, the misinterpreted part was not mixed with the actual flow, so was easily removed without affecting the part of the image that represented the actual flow body. The flow was assumed to be larger than the misinterpreted parts. Thus, if more than a single white region was identified by the FCNN as the flow, only the largest region was preserved, and the other parts were removed.

The PIV algorithm is based on a different principle than semantic segmentation algorithms. Specifically, the PIV algorithm tracks the texture of flow and assumes that the velocity of the tracers represents the flow. The flow material in the field-scale experiment contained gravel, which served as tracers [30] to generate flow textures for PIV analysis. However, the low resolution of the images makes it difficult to detect individual particles of gravel. Moreover, gravel particles may saltate and not always have a velocity that is representative of that of the flow body. So, it is not surprising to find that PIV may underestimate the flow velocity. Specifically, images M to P show that the flow has a velocity that is less than 1 m/s, whereas the measured ground truth [23] had a flow velocity of 5.2 m/s.

Figure 10 compares the performances of the different algorithms in computing the movement of the flow head in the field-scale experiment. For the TWS and FCNN methods, the travel distance was defined as the distance from the gate of the container to the flow head. Through semantic segmentation, the pixels of the flow head were tracked in each frame, and the positions of the pixels were used to compute the time history of the travel distance. In the segmentation results produced by the FCNN (i.e., the first row in Fig. 9), the largest continuous white area was the flow, and the pixel in this area that was farthest from the gate was considered the flow head. Evidently, the pixel position computed by the FCNN (solid line in Fig. 10a) closely approximated the measurements in the experimental study [23]. However, TWS produced noisy results and struggled to identify the flow and define the flow head. For example, three solid lines corresponding to the 50, 70, and 90% percentiles of the travel distance of the identified flow pixels are presented in the figure. For a noise-free segmentation result, the 100% percentile of the travel distance (i.e., the maximum travel distance of all pixels) should be the travel distance of the flow head. In noisy results, however, some pixels identified as the flow are misinterpreted pixels. As such, using a high percentile (e.g., corresponding to the 70 and 90% percentiles of the travel distance) includes misinterpreted pixels in the computed travel distance. Thus, the travel distance is overestimated. In contrast, if a low percentile of the pixel

travel distance (e.g., corresponding to 50%) is used, the flow travel distance is underestimated and some of the pixels corresponding to the actual flow are excluded.

The PIV algorithm does not directly provide information on the travel distance. However, the velocity field produced by PIV can be used to differentiate moving from static objects. In a flow image, the flow can be regarded as a moving object and the background is static. By setting a velocity threshold, the boundaries of moving objects can be identified. Figure 10b shows the results of using this strategy to identify the flow body and its head. Four velocity thresholds (i.e., 0.1, 0.4, 0.7, and 1.0 m/s) were used. In some frames of the flow video, the travel distances computed with the strategy were close to those measured in the experimental study [23]. For example, the frames between 1.0 and 1.5 s are plotted close to the red markers indicating the travel distances measured in the experimental study [23]. However, the PIV algorithm overestimated the flow runout in many other frames because some static parts of the flow near the downstream end of the flume were misinterpreted as moving objects owing to the limited image resolution and a lack of tracers (e.g., the PIV image at 2.04 s in Fig. 9).

4.4 Evaluating laboratory-scale flow boundaries from the side

High-resolution images from the sides of laboratory-scale models have been analyzed in numerous PIV studies [8, 10, 30]. The main limitation of PIV is that it relies heavily on tracer-generated textures. For example, Fig. 11 shows the velocity field of a submarine debris flow deduced with PIV. Although tracers have been added to the flow to generate textures and facilitate PIV analysis, the tracers do not always move with the flow body. Specifically, there are no tracers in the diluted parts of the flow (e.g., areas bounded by dashed yellow lines in Fig. 11b and c). So the velocity field in this part of the flow is absent or not correctly computed. If an analysis of the flow height is required, the missing vectors in the diluted areas may cause an underestimation. Also, the reference grids on the side walls (Fig. 11a) may cause an underestimation of the overall kinematics in PIV. For the flow body around a reference grid, velocities deduced by PIV are close to zero (e.g., the areas bounded by the dashed red lines in Fig. 11b and c).

Owing to missing or underestimated velocity fields, PIV is limited in identifying moving objects or flow boundaries. For example, the green contours in Fig. 11 were identified using the thresholding strategy described in Sect. 3 with a threshold of 4.5 m/s. Although this threshold is smaller than the minimum flow frontal velocity reported in the experimental study [10], it does not allow the identification

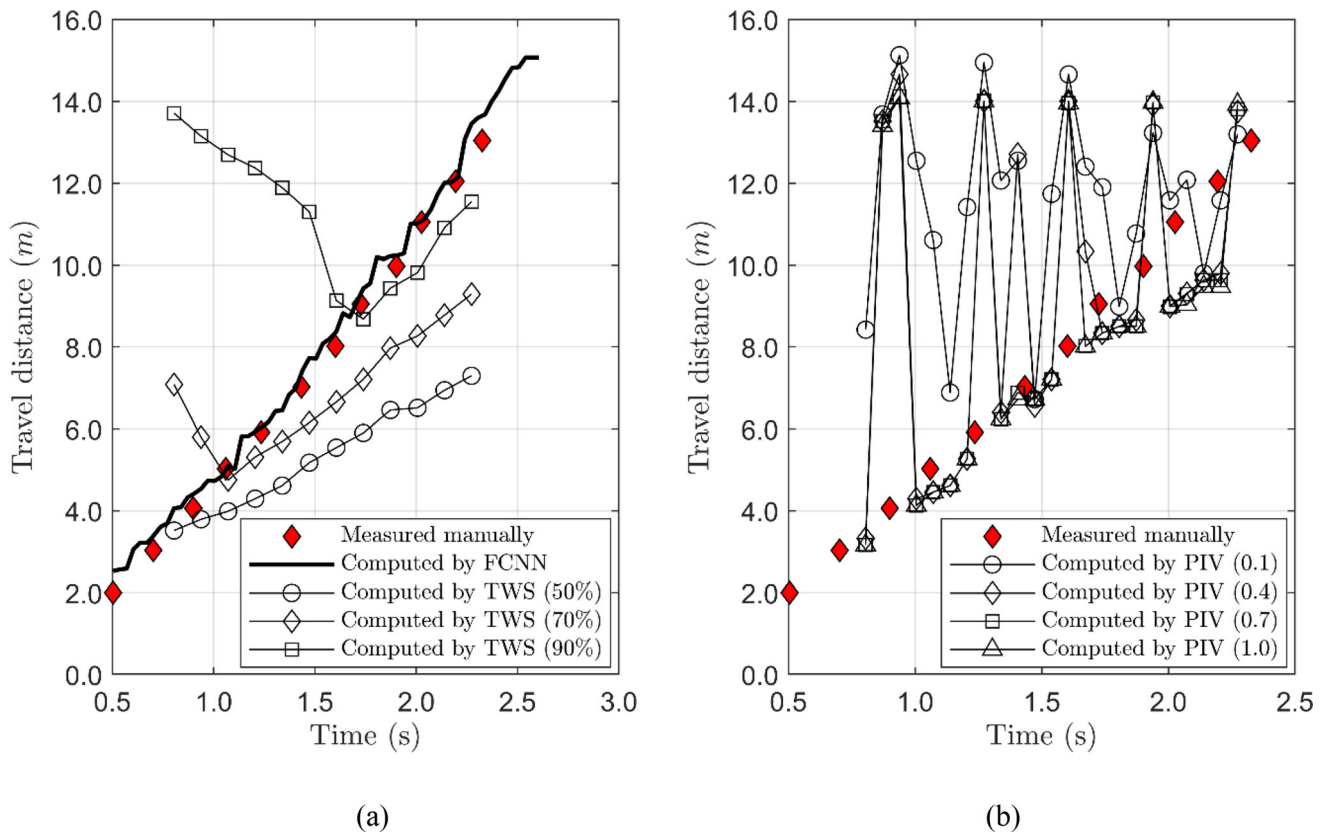


Fig. 10 Comparison of the time histories of the measured [23] and computed movements of the flow head. (a) Results of semantic segmentation. The movement of the flow head computed by TWS represents the 50, 70, and 90% percentiles of the distance of flow pixels from the gate. (b) Results of PIV (based on a velocity threshold)

of a complete flow front. Moreover, even if a small threshold (e.g., 0.15 m/s corresponding to the red contours) is used, areas around reference grids and diluted flows cannot be identified. In contrast, the FCNN yielded a complete flow boundary identification result throughout the flow process (i.e., the purple dashed line). Although the FCNN does not produce a velocity field, the FCNN outperformed PIV in tracking the flow front and flow height even after tracers were added to facilitate PIV analysis.

4.5 Tracking flow boundaries of unchanneled flow–forest interaction

The tracking of flow boundaries is useful for unchanneled flows [21]. Figure 12 compares the ability of the FCNN with thresholding and TWS in carrying out semantic segmentation for complex flow–forest interactions. Compared with channeled flows, which were limited to a width of 0.2 m [10], unchanneled flows spread almost 2 m in width in the unchanneled experiment, making it difficult to ensure uniform light conditions when a flow spreads over a large area. Complex shadows and specular reflections in flow images may hinder semantic segmentation. Moreover,

tracking flow boundaries becomes more difficult when obstacles are involved because obstacles can occlude the flow body or be misinterpreted as part of the flow. Such effects introduce errors into semantic segmentation, especially when a large number of obstacles are involved. For cases without obstacles, the major challenges are specular reflections and shadows (Fig. 12a and b). When using a straightforward thresholding method, specular highlights are misinterpreted as the background, and shadows are misinterpreted as the flow because thresholding uses color brightness to differentiate between the flow and background. The brown flow is darker than the white basal board, and thresholding thus identifies any dark part (i.e., the shadows) as the flow and any bright part (i.e., the highlights) as the background. In contrast with the thresholding method, TWS uses a more sophisticated criterion for differentiating between the flow and background (refer to [2]). As such, TWS produces results that are less noisy and correctly excludes shadows from the flow. However, noise is still observed. In contrast, the FCNN uses self-adaptive criteria based on training, which generates clean results with little noise and boundaries that closely approximate the observed boundaries.

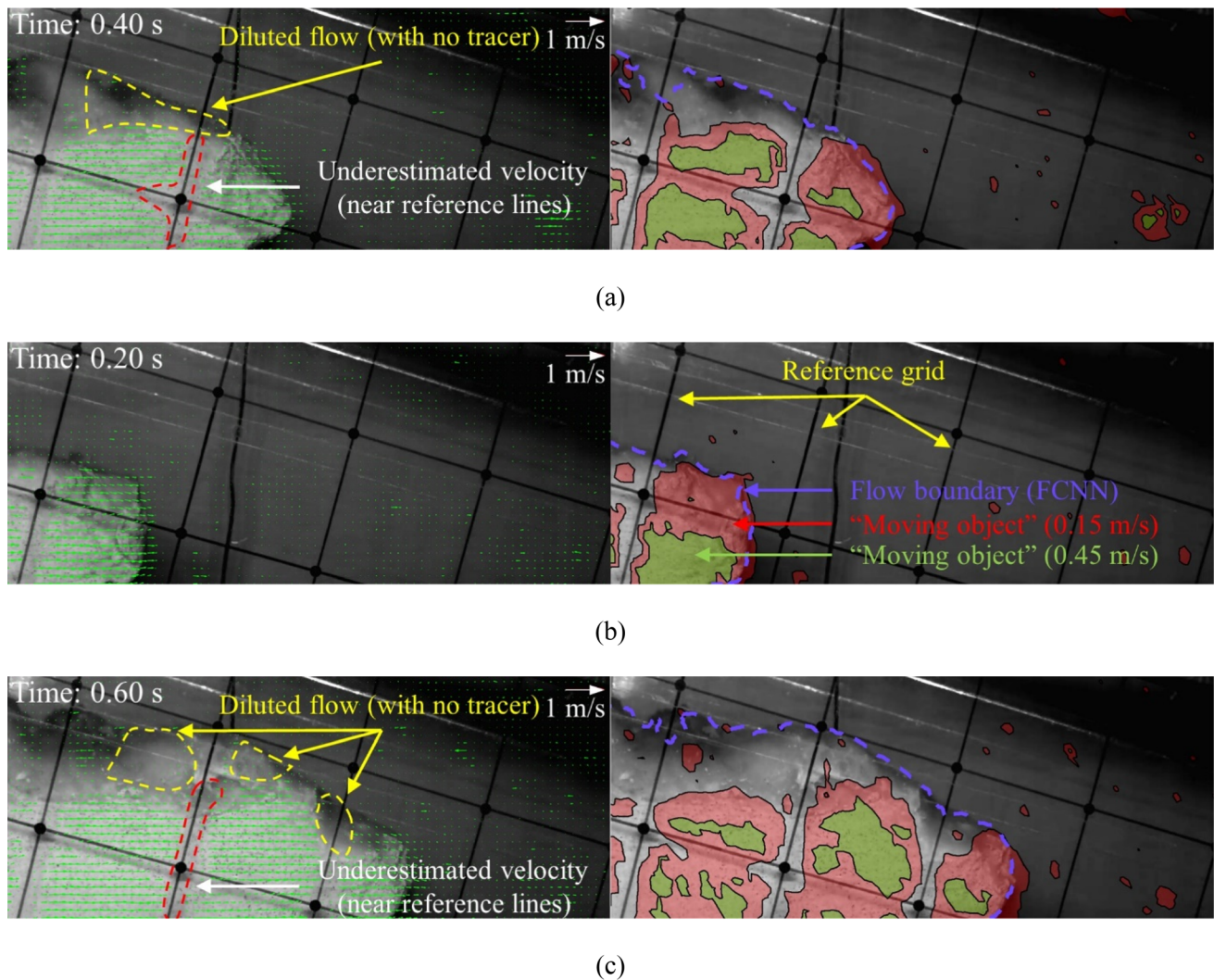
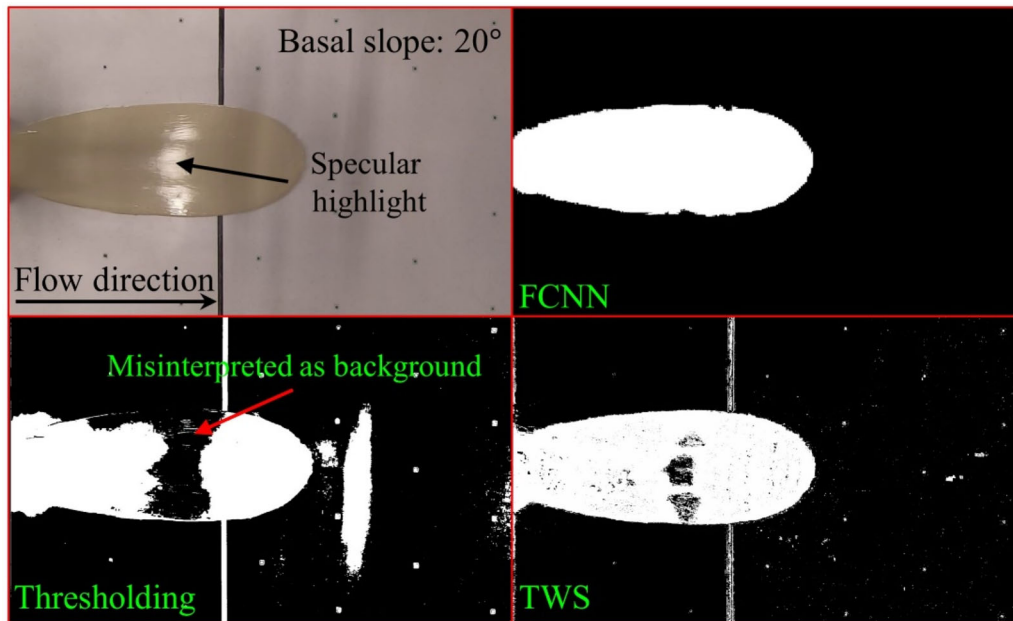


Fig. 11 Computed velocity fields and boundaries of submarine flows at different times after the flow entered the FOV (as described in Fig. 3): (a) 0.2 s, (b) 0.4 s, and (c) 0.6 s. The green arrows show the velocity field computed through PIV. The red and green contours were identified as “moving objects” using thresholding PIV data. The dashed purple line is the flow boundary identified using the FCNN

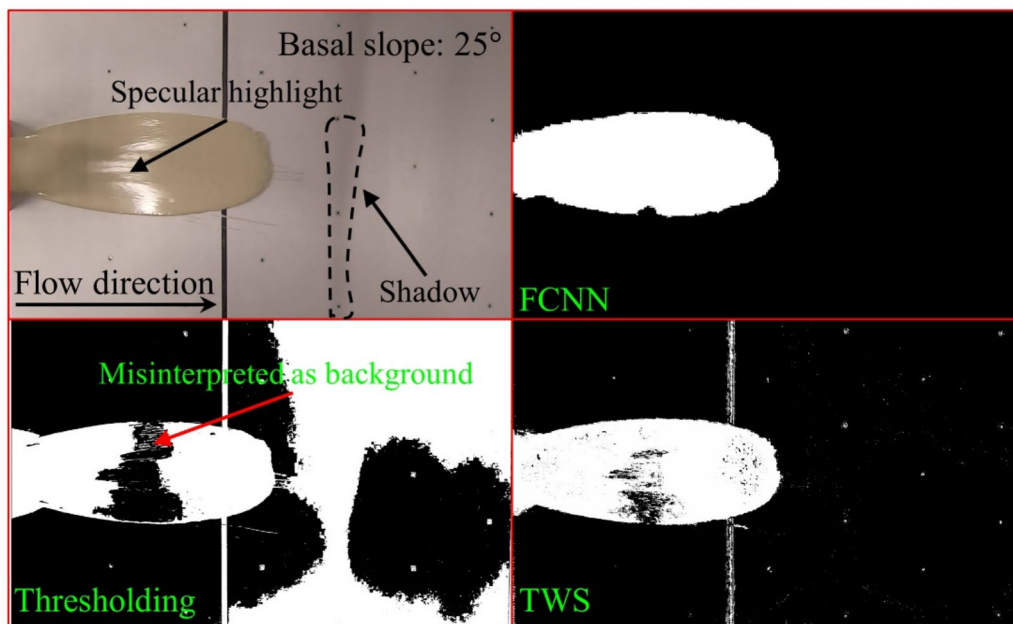
For unchanneled flows, although the results of TWS and thresholding are noisy, they can still be used to extract flow boundaries if manual post-processing is adopted (e.g., manually excluding the misinterpreted parts of the imagery using ImageJ [29]). However, if obstacles are involved, it is difficult to improve segmentation results even with manual post-processing. For example, Figs. 12c and 13d compare segmentation results for experiments with a total of 116 obstacles. The aluminum obstacles were darker than the white background and thus misinterpreted as part of the flow. The flow and misinterpreted parts were blended in the semantic segmentation results of thresholding and TWS. It is thus impractical to separate such misinterpreted areas from the real flow parts. In contrast, the FCNN differentiated between obstacles and flows and produced clean segmentation results.

5 Discussion

The framework proposed in this study harnesses the power of FCNN to process flow images and thus allows modelers to extract valuable information of flow kinematics from both new and existing physical experiments. In previous experiments, such as those conducted in [23, 34], although flow images are captured, the interpretation of these images often relied on human operation or image processing tools like ImageJ or Photoshop. Such manual interpretation process can be labor-intensive and time-consuming, leading to limitations in the number of sampling points of flow kinematics. For instance, in [23], only less than 15 scattered data points were used to represent the time history of flow travel distance in a large-scale flume experiment. This

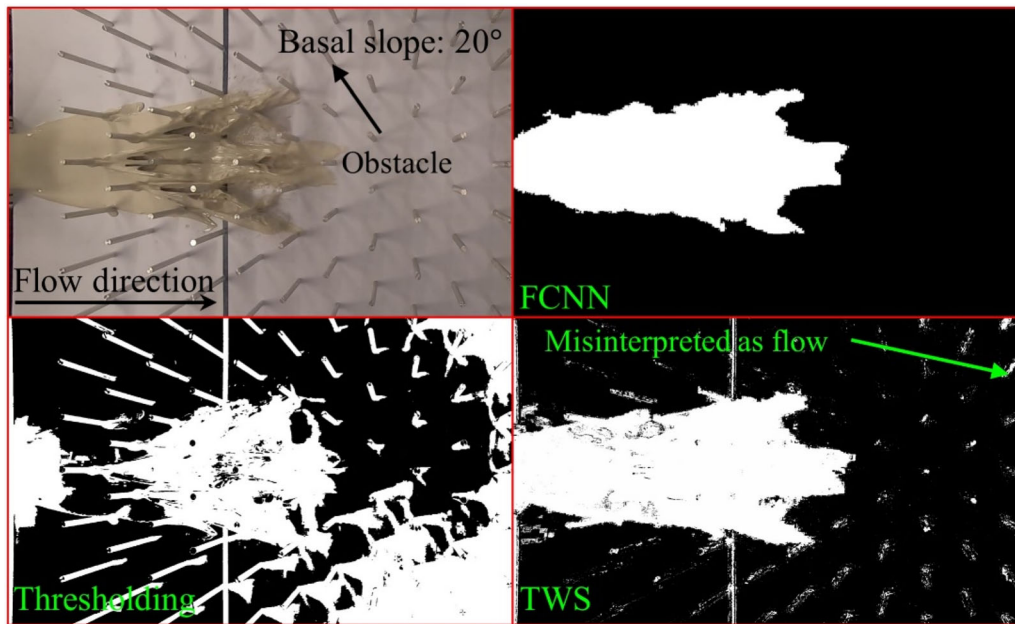


(a)

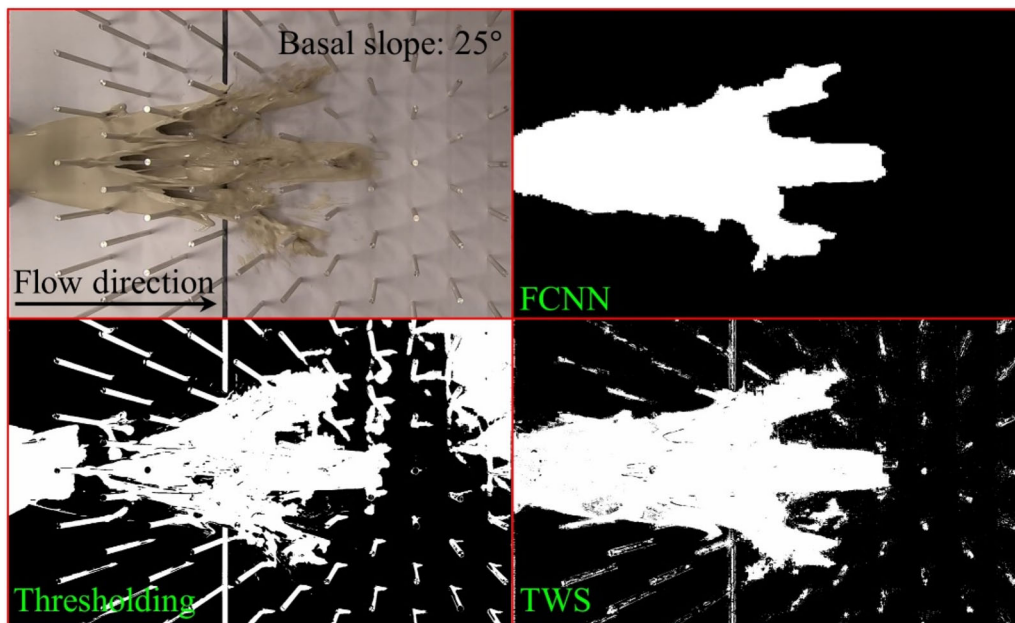


(b)

Fig. 12 Comparison of FCNN segmentation results of flow–forest interactions with thresholding and TWS: (a) on a clear board with a basal slope of 20° , (b) on a clear board with a basal slope of 25° , (c) on a board with 116 model trees and a basal slope of 20° , and (d) on a board with 116 model trees and a basal slope of 25° . Automated thresholding was performed via ImageJ [29]. TWS was carried out following the procedures reported in the literature [2]



(c)



(d)

Fig. 12 continued

scarcity of data points can be attributed to the manual measurement process, which required the physical modelers to analyze the recorded flow videos manually and measure the flow travel distance by hand. In contrast, the use of FCNN provides an automated solution for image processing. This enables researchers to obtain continuous

time histories of flow kinematics, including the flow behavior in the gaps between the limited sampling points (as shown in Fig. 10). By leveraging FCNN's capabilities, the study offers a method to make full use of expensive physical experiments. Additionally, the continuous time history of flow kinematics can be a benchmark for

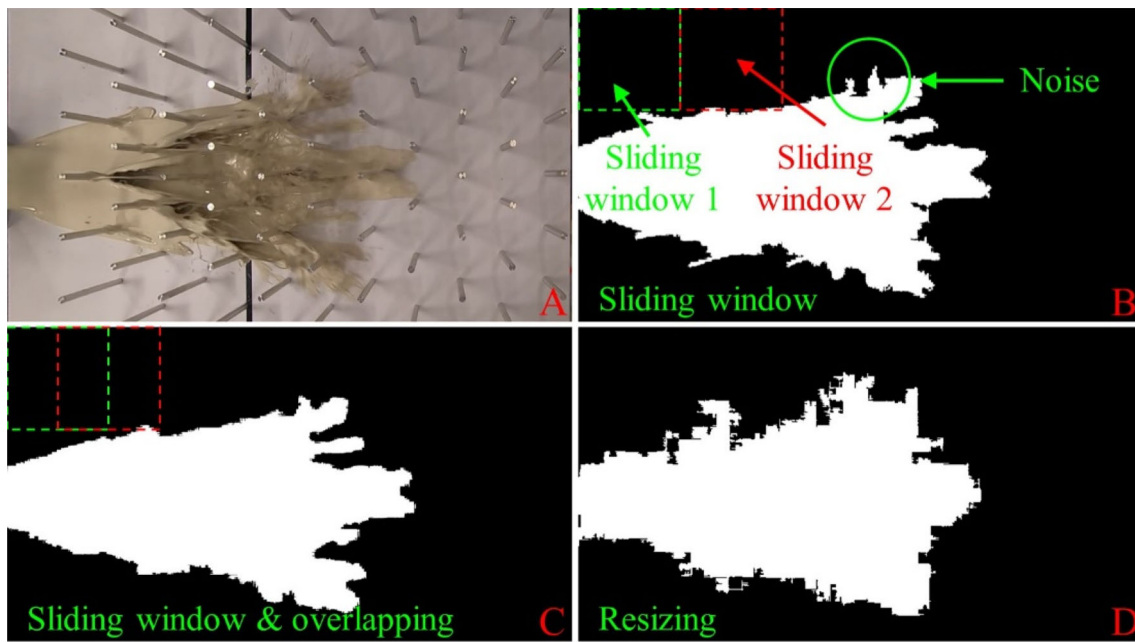


Fig. 13 Comparison of preprocessing techniques for large image segmentation using FCNN: Image A is the Input flow image. Image B illustrates the noisy segmentation result obtained through the naive sliding window technique. Image C shows the enhanced segmentation result by incorporating overlaps between sliding windows. Image D shows the segmentation result obtained by resizing the input image

calibrating numerical simulations and building high-fidelity numerical tools as demonstrated in [18].

When training the FCNN model, it is worth noting that the FCNN model employed in this study is a variant of U-net architecture [27]. The U-net architecture is known for its high accuracy even with a limited number of training images [19]. This characteristic is particularly valuable considering that annotating experimental training images can be time-consuming, especially when dealing with high-speed cameras that capture thousands of frames per second. In Fig. 7, the model achieved a validation accuracy of 97% using fewer than 15 annotated images. However, it is important to note that the generalization ability of the FCNN model, particularly in out-of-distribution scenarios, is limited. This limitation arises because deep learning models may suffer from overfitting [37] when the training images do not cover a wide range of experimental conditions, such as location, timing, texture, and camera settings. Therefore, it is advised to not assume that an FCNN trained with one set of experimental images will be applicable to other sets of experiments. Nonetheless, the results in Sect. 4 demonstrate that the FCNN successfully processes an entire group of experiments, even when only a limited number of images (up to 15 in this case) from those experiments were used for training. This success can be attributed to the similarity of experimental conditions within the same group of experiments. In situations where physical modelers need to analyze a new and different group of experiments, the FCNN can be retrained to

digitize the new data. However, the time cost associated with annotation and retraining is relatively trivial compared to the planning, execution, setup of cameras, and manual processing of videos using traditional image processing methods.

When using the FCNN for flow image segmentation, a sliding window technique, as described in Fig. 1, was invoked to preprocess the input image. The primary objective of this technique was to efficiently process images of any size with limited GPU memory. In cases where the images exceeded the GPU memory capacity, the flow images were processed in parts. However, it is important to note that the sliding window technique may introduce noise in the segmentation results, as demonstrated in Fig. 13. Specifically, in image B, the sliding window approach discards the contextual information of the entire image during the segmentation process. Consequently, the flows are solely processed based on the cropped image patches. In the central region of the sliding windows, the FCNN can still generate accurate segmentation results because there is sufficient nearby context. However, difficulties arise when processing images near the sliding window boundaries, leading to the generation of noise in those areas. To mitigate the noise, this study incorporates overlapping between sliding windows. As illustrated in image C, a 50% overlap is ensured for each sliding window during the sliding process, and the final segmentation result is determined by averaging the results from different sliding windows. The overlapping approach allows each pixel on the flow image

to be processed multiple times, thereby improving the reliability of the results. This explains why image C exhibits smoother flow boundaries compared to image B. However, it is worth noting that the sliding window technique requires invoking the FCNN multiple times, and the overlapping technique doubles the number of FCNN segmentations performed, resulting in reduced computational efficiency. An alternative approach could involve resizing the large flow image to match the required resolution for the FCNN input. After segmentation, the produced masks can be upscaled back to the original resolution. However, this technique should be used with caution and only when the flow image is slightly larger than the FCNN input resolution. Image D presents a failure case of using resizing, where the flow image is downscaled by a factor of 5.5 on both width and height to fit the FCNN input resolution. The segmentation accuracy is significantly lower compared to the sliding window technique. This reduction in accuracy is not surprising because downscaling discards information from the flow image, and such information cannot be fully recovered during the upscaling of masks.

Figure 10a shows that the modified U-net model offers precise estimation of flow travel distance. Nevertheless, given the swift progression of machine learning algorithms, future physical modelers may benefit from adopting more advanced segmentation models such as the Swin transformer [20] or Segment Anything [17] to achieve greater segmentation accuracy, albeit at the expense of building a more sophisticated software framework with higher memory requirements. Nonetheless, the proposed workflow can serve as a useful reference when constructing such a software framework.

6 Conclusions

A new framework for digitalizing experimental flow-type landslide kinematics was proposed in this study. The FCNN model was evaluated using different types of experiments. Key conclusions are as follows:

- (1) Adopting deep learning, images of flow obtained in experiments were processed to obtain the boundaries of the flow head so that the flow kinematics could be digitalized. The proposed method outperformed classic image processing algorithms (i.e., PIV, TWS, and thresholding algorithms) and had an accuracy in boundary identification close to that of visual observation.
- (2) With image correction, the FCNN model can process images from consumer-grade cameras. This feature is most useful for measurements in outdoor field-scale experiments and measurements of real flow events.

- (3) With fewer than 15 annotated training images, the FCNN digitalized experimental flows with an accuracy of 97% in semantic segmentation. The FCNN can process images under complex light and shadow conditions and thus be used in experiments where a plain background and uniform light conditions are difficult to prepare.

Appendix A: architecture of the FCNN model

Figure 14 shows the architecture of the modified U-net. The architecture follows an encoder–decoder style, where an input image is passed to a set of encoding blocks (i.e., the blue rectangles) and then to a set of decoding blocks (i.e., the orange rectangles). An image that is passed to encoding blocks can be regarded as a three-dimensional $720 \times 720 \times 3$ matrix (i.e., the light blue stripe). In the first encoding block, an image is processed through convolution, batch normalization, and max-pooling operations [37, 38]. After these operations are performed, the input matrix is converted into another three-dimensional matrix with dimensions of $360 \times 360 \times 16$. This new three-dimensional matrix is referred to as a feature map, and the size of the third dimension (i.e., 16) is referred to as the number of channels. The aforementioned operations that transform an input image into a feature map are analogous to filtering [26] in classic digital image processing (DIP). The main difference between the operations conducted by the FCNN and filtering in classic DIP is that the operators used in classic DIP are artificial operators (e.g., the canny operator [4] designed for edge detection). Artificial operators are designed according to assumptions, and there are inevitably input images that are not suitable because of the assumptions made. As such, classic DIP may not perform well in identifying the boundaries in flow images with complex ambient light and shadows (as discussed in Sect. 3). In contrast, the operators used in the convolution operation of the FCNN were obtained by training. Therefore, well-trained operators are suitable for the input images used.

There are four encoding blocks in the encoder. Each encoding block is connected to the next one. The input of an encoding block comprises two parts: (1) the output from the previous block and (2) the down-sampled input of the previous block (i.e., the gray arrows). Taking the first encoding block as an example, the input of the block is first down-sampled in a special convolution operation, which directly converts the input (i.e., a $720 \times 720 \times 3$ matrix) as a special feature map that has the same size as the output of the block (i.e., a $360 \times 360 \times 16$ matrix). The special feature map and output of the block are then merged to

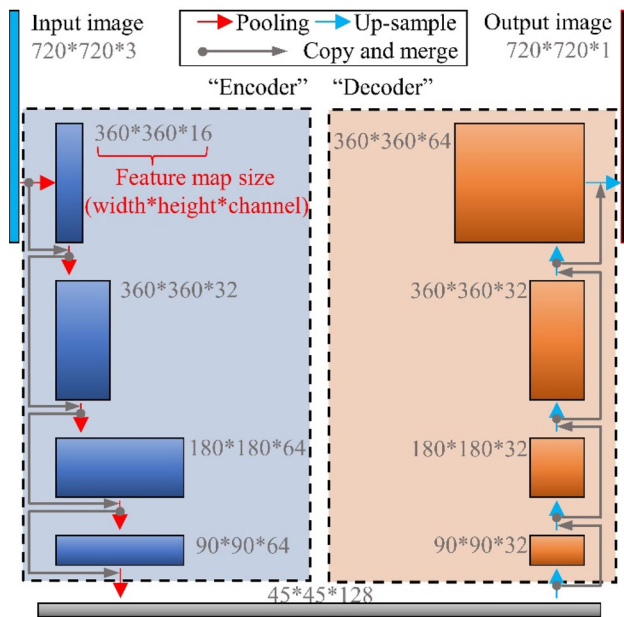


Fig. 14 Schematic of the FCNN architecture

produce a new $360 \times 360 \times 16$ matrix. The newly merged matrix is the input of the next encoding block. The merging operation is analogous to building shortcuts between blocks. Such a shortcut strategy was first introduced in the well-known residual neural network [14]. The shortcuts enable encoding blocks to use more information than just the output from the previous block. Consequently, the information loss due to the convolution and max-pooling operations in the previous blocks is reduced to facilitate the training of the deep FCNN.

Acknowledgements The authors are grateful for financial support provided by the General Research Fund of the Hong Kong Research Grants Council (17206622).

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors state that they have no competing financial interests or personal relationships that could have influenced the work reported in this paper.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted

use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Alvarez L, Gomez L, Sendra R (2010) Algebraic lens distortion model estimation. *Image Processing Online* 1
- Arganda-Carreras I, Kaynig V, Rueden C, Eliceiri KW, Schindelin J, Cardona A, Seung HS (2017) Trainable Weka Segmentation: a machine learning tool for microscopy pixel classification. *Bioinformatics* 33(15):2424–2426
- Benito J, Bertho Y, Ippolito I, Gondret P (2012) Stability of a granular layer on an inclined “fakir plane.” *EPL* 100(3):34004
- Canny J (1986) A computational approach to edge detection. *IEEE Computer Society*
- Chen Z, Rickenmann D, Zhang Y, He S (2021) Effects of obstacle's curvature on shock dynamics of gravity-driven granular flows impacting a circular cylinder. *Eng Geol* 293:106343
- Choi CE, Ng CWW, Song D, Kwan J, Shiu H, Ho KKS, Koo RC (2014) Flume investigation of landslide debris-resisting baffles. *Can Geotech J* 51(5):540–553
- Chollet F (2019) Keras guides: image segmentation with a U-Net-like architecture. <https://github.com/keras-team/keras-tuner>
- Cui X, Harris M, Howarth M, Zealey D, Brown R, Shepherd J (2022) Granular flow around a cylindrical obstacle in an inclined chute. *Phys Fluids* 34(9):093308
- de Haas T, Braat L, Leuven J, Lokhorst IR, Kleinhans MG (2015) Effects of debris flow composition on runout, depositional mechanisms, and deposit morphology in laboratory experiments. *J Geophys Res Earth Surf* 120(9):1949–1972
- Du J, Choi CE, Yu J, Thakur V (2022) Mechanisms of submarine debris flow growth. *J Geophys Res Earth Surf* 127(3):e2021JF006470
- Fabijanska A (2018) Segmentation of corneal endothelium images using a U-Net-based convolutional neural network. *Artif Intell Med* 88:1–13
- Gonzalez RC, Woods RE, Eddins SL (2004) *Digital image processing using MATLAB*. Pearson Prentice Hall, New Jersey
- Goodwin GR, Choi CE (2022) A depth-averaged SPH study on spreading mechanisms of geophysical flows in debris basins: implications for terminal barrier design requirements. *Comput Geotech* 141:104503
- 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
- Iverson RM (1997) The physics of debris flows. *Rev Geophys* 35(3):245–296
- Kingma D.P., Ba J (2014) Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*
- Kirillov A, Mintun E, Ravi N, Mao H, Rolland C, Gustafson L, Xiao T, Whitehead S, Berg AC, Lo W-Y (2023) Segment anything. *arXiv preprint arXiv:2304.02643*
- Liang Z, Choi CE, Zhao Y, Jiang Y, Choo J (2023) Revealing the role of forests in the mobility of geophysical flows. *Comput Geotech* 155:105194
- Liang Z, Nie Z, An A, Gong J, Wang X (2019) A particle shape extraction and evaluation method using a deep convolutional neural network and digital image processing. *Powder Technol* 353:156–170
- Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B (2021) Swin transformer: Hierarchical vision transformer using

- shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 10012–10022
21. Luong TH, Baker JL, Einav I (2020) Spread-out and slow-down of granular flows through model forests. *Granul Matter* 22(1):10
 22. Maas H, Gruen A, Papantoniou D (1993) Particle tracking velocimetry in three-dimensional flows. *Exp Fluids* 15(2):133–146
 23. Ng CWW, Liu H, Choi C, Kwan J, Pun W (2021) Impact dynamics of boulder-enriched debris flow on a rigid barrier. *J Geotech Geoenviron Eng* 147:04021004
 24. Pham M-V, Kim Y-T (2022) Debris flow detection and velocity estimation using deep convolutional neural network and image processing. *Landslides*. <https://doi.org/10.1007/s10346-022-01931-6>
 25. Pi Y, Nath ND, Behzadan AH (2020) Convolutional neural networks for object detection in aerial imagery for disaster response and recovery. *Adv Eng Inform* 43:101009
 26. Pratt WK (2013) Introduction to digital image processing. CRC press
 27. Ronneberger O, Fischer P, Brox T (2015) U-Net: convolutional networks for biomedical image segmentation. In: Navab N, Hornegger J, Wells WM, Frangi AF (eds) *Medical Image Computing And Computer-Assisted Intervention*, Cham 2015. Lecture Notes in Computer Science, Springer, pp 234–241
 28. Russell BC, Torralba A, Murphy KP, Freeman WT (2008) LabelMe: a database and web-based tool for image annotation. *Int J Comput Vis* 77(1–3):157–173
 29. Schindelin J, Arganda-Carreras I, Frise E, Kaynig V, Longair M, Pietzsch T, Preibisch S, Rueden C, Saalfeld S, Schmid B (2012) Fiji: an open-source platform for biological-image analysis. *Nat Methods* 9(7):676–682
 30. Song P, Choi CE (2021) Revealing the Importance of capillary and collisional stresses on soil bed erosion Induced by debris flows. *J Geophys Res Earth Surf* 126(5):e2020JF005930
 31. Taylor-Noonan A, Gollin D, Bowman E, Take W (2021) The influence of image analysis methodology on the calculation of granular temperature for granular flows. *Granul Matter* 23(4):96
 32. Thielicke W, Stamhuis EJ (2014) PIVlab – towards user-friendly, affordable and accurate digital particle image velocimetry in MATLAB. *J Open Res Softw*. <https://doi.org/10.5334/jors.bl>
 33. Vicari H, Ng CWW, Nordal S, Thakur V, De Silva R, Liu H, Choi C (2021) The effects of upstream flexible barrier on the debris flow entrainment and impact dynamics on a terminal barrier. *Can Geotech J*. <https://doi.org/10.1139/cgj-2021-0119>
 34. Wang F, Chen XQ, Chen JG (2017) Experimental study on the energy dissipation characteristics of debris flow deceleration baffles. *J Mt Sci* 14(10):1951–1960
 35. White DJ, Take WA, Bolton MD (2003) Soil deformation measurement using particle image velocimetry (PIV) and photogrammetry. *Geotechnique* 53(7):619–631
 36. Wu L, Shang Q, Sun Y, Bai X (2019) A self-adaptive correction method for perspective distortions of image. *Front Comput Sci* 13(3):588–598
 37. Bengio Y, Goodfellow I, Courville A (2016) Deep learning. MIT Press
 38. Zhan HJ, Chen GL, Lu Y (2016) Applying batch normalization to hybrid NN-HMM model for speech recognition. In: Tan T, Li X, Chen X, Zhou J, Yang J, Cheng H (eds) *Pattern Recognition*, vol 663. Communications in Computer and Information Science, Springer, Singapore, pp 427–435

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.