Sensors and Materials, Vol. 37, No. 6 (2025) 2463–2487
MYU Tokyo

S & M 4067

Debris Pattern Recognition Based on Visual Sensor and Image Stitching Technology

Wei-Yuan Zheng and Jih-Gau Juang*

Department of Communications, Navigation and Control Engineering, National Taiwan Ocean University
2 Pei-Ning Road, Keelung 20224, Taiwan

(Received January 15, 2025; accepted June 5, 2025)

Keywords: image stitching, object detection, deep learning

In this study, we applied multiple unmanned aerial vehicles (UAVs) and visual sensors with deep learning neural networks, You Only Look Once (YOLO), to quickly and effectively recognize significant areas of debris. Information on debris locations, area sizes, and images is sent to the monitoring system. Then, the debris distribution is analyzed, and the source of the debris can be found. The pattern recognition process uses a variety of feature detection methods, description, and point matching for real-time image stitching of the scene. The UAVs can obtain large-area scene images and check whether undetected debris exists. A comparison with different YOLO models is given. The effects of debris recognition and the consequences of various types of data and image stitching during the image stitching process are applied to analyze the real-time image stitching effects by different methods.

1. Introduction

Many cities have rivers of different sizes. Bridges are built in the river environment, so there are numerous banks, embankments, and piers. Even with laws and regulations, it is difficult to curb the problem of littering. For the preservation of the living environment and ecological protection, we use a real-time unmanned aerial vehicle (UAV) waste monitoring system[(1)] with artificial intelligence to recognize debris and stitch river scene images so that government officials can effectively and clearly understand the area of debris status and make efforts to control the source of debris. We developed a real-time UAV waste monitoring system that uses the You Only Look Once (YOLO) object detection architecture,[(1)] and we also created the Haida garbage dataset,[(2)] which can provide training for the YOLO dataset. The system can be used in different scenes, and the identified debris can be processed using a cloud site as the source of real-time image stitching. Kumareswaran *et al.* used drones to perform image stitching in near real time[(3)] and compared various feature detectors such as binary robust invariant scalable keypoint (BRISK), scale-invariant feature transform (SIFT),[(4)] unmodified SIFT, oriented features from accelerated segment test (FAST), rotated binary robust independent elementary features (BRIEF) (ORB),[(5)] and accelerated-KAZE (AKAZE).[(6)] They then used the image

*Corresponding author: e-mail: jgjuang@mail.ntou.edu.tw
https://doi.org/10.18494/SAM5557

ISSN 0914-4935

stitching process and compared the calculation speed and qualitative evaluation. They concluded that it is most suitable to use ORB and AKAZE for real-time image stitching because the calculation time is short. In Ref. 7, the theoretical foci of various feature detection and description methods are listed, and their performances are compared. In Ref. 8, the shortcomings of AKAZE are pointed out, method improvements are proposed, and the feature extraction efficiency and matching accuracies of SIFT, ORB, AKAZE, and the improved AKAZE are compared. In Ref. 9, two types of feature detection and description were used, SIFT and speeded-up robust features (SURF), and different methods for eliminating mismatch points, such as random sample consensus (RANSAC) and M-estimate sample consensus (MASC), were used for comparison. The conclusion is that SURF has better computation time and matching accuracy than SIFT. SURF is compared with MSAC and RANSAC, and it is found that MSAC is the most stable and fastest, and the matching accuracy is the highest. Wang *et al.* built a deep learning model of YOLOv7 by improving the previous YOLO architecture and model scaling concepts, and compared the data with those of other YOLOs, showing that it is fast and accurate.[10]

To accurately obtain the UAV's Global Navigation Satellite System (GNSS) coordinate position and direction angle, refer to the research of Wang,[11] who used the same two sets of GNSS receivers and real-time kinematic (RTK) technology, with which the error of the coordinate position can be obtained at the centimeter level. The high-precision coordinate position enables the acquisition of an accurate direction angle of the UAV. Combined with the above method, we improved the UAV GNSS coordinate accuracy, calculated a stable direction angle, then used a real-time UAV waste monitoring system with YOLOv5 and YOLOv7 for debris recognition, and used different feature detection and description methods and feature identification point matching for dynamic real-time image stitching. The real-time UAV waste monitoring system[1] was applied in this study. This system is based on YOLO for object detection; YOLOv5 and YOLOv7 are used for debris recognition, and the training status and identification results are compared. This system also achieves UAVs' real-time dynamic image stitching, mainly through feature detection and description, feature matching, and homography matrix computation. Feature detection and description is accomplished with SIFT, ORB, and AKAZE. A brute-force matcher with K-nearest neighbors (K-NN) and a fast library for approximate nearest neighbors (FLANN)-based matcher with K-NN matching are used for feature matching. The homography matrix can use RANSAC and progressive sample consensus (PROSAC) to eliminate mismatches. Various combinations of the above methods yield 12 image stitching methods, the results of which can be compared.

2. System Design

The real-time UAV waste monitoring system consists of several components and sensors and is divided into nine parts. The sender and the receiver are set up on the laboratory computer server, except for UAV images and data. The images are transmitted to the Video Streaming Server, kafka, as a real-time data connection. mongoDB is used for data storage and real-time monitoring, which can be displayed on the website. Figure 1 shows the monitoring system. The

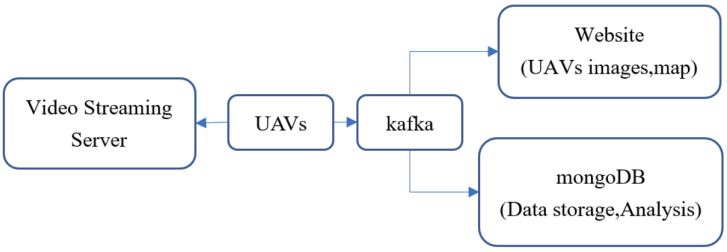


Fig. 1. (Color online) Architecture of a real-time UAV waste monitoring system.

sender uses the embedded system and 4G network to send real-time images, UAV's GNSS coordinates, and garbage information to the server. Real-time image stitching can also be done through this system. Figure 2 shows the architecture of the UAV sender.

The embedded system and 12 V battery are afixed on the UAV. A 4G dongle provides the network connection to the embedded system to send the UAV's GNSS coordinates, debris identification results, size of area, and real-time images to the laboratory computer server. The video equipment includes a USB video class (UVC) webcam such as the Microsoft LifeCam Studio,[12] and it is loaded on the 2D gimbal of the UAV to maintain the stability of the video image. The image transmission method is imageZMQ,[13] and the rest of the data is streamed through kafka. The embedded process system uses NVIDIA Jetson Xavier NX, which has the advantages of small size, light weight, and high performance. It can be used on UAVs for garbage identification, data and image reception, and transmission.[14,15] Pixhawk 6X is used for the flight control of the hexacopter (Fig. 3). Peripheral equipment includes a receiver, telemetry radio, GNSS, motor, electronic speed controller (ESC), and battery; the specifications and quantities of the motors and ESCs are selected in accordance with the requirements.

Pixhawk 6X's FMU processor runs at 480 MHz with 2 MB of flash memory and 1 MB of RAM; the IO processor runs at 72 MHz with 64 KB SRAM. Compared with previous generations of Pixhawk, its performance has been improved, and it has triple IMU sensors with a low-noise double barometer. When the Pixhawk autopilot detects a sensor failure, it seamlessly switches to other sensors to maintain flight stability. It adopts an entirely isolated sensor domain, each with an independent bus and power control. Its vibration isolation system can filter and eliminate high-frequency vibration and reduce noise to ensure accurate signal readings, enabling UAVs to perform better flights.[16]

The UAV is equipped with two GNSS receivers. It has a dual-frequency GNSS antenna, an electronic compass, and a low-cost, high-precision dual-frequency GNSS receiver chip, U-blox F9P, which uses real-time kinematic technology to obtain accurate GNSS coordinates. The direction angle of the UAV can be obtained from two sets of high-precision GNSS coordinate positions. The GPS for the yaw setting reference of the UAV is shown in Refs. 17 and 18. The UAV uses Sik telemetry radio v3[19] as the ground station to communicate with the flight controller in real time and obtain various data about the UAV, such as altitude (m), ground speed (m/s), current position, and UAV warning signals. The two hexacopters used are equipped with a 433 MHz telemetry radio. The telemetry radio is not a one-to-one device, so there may be

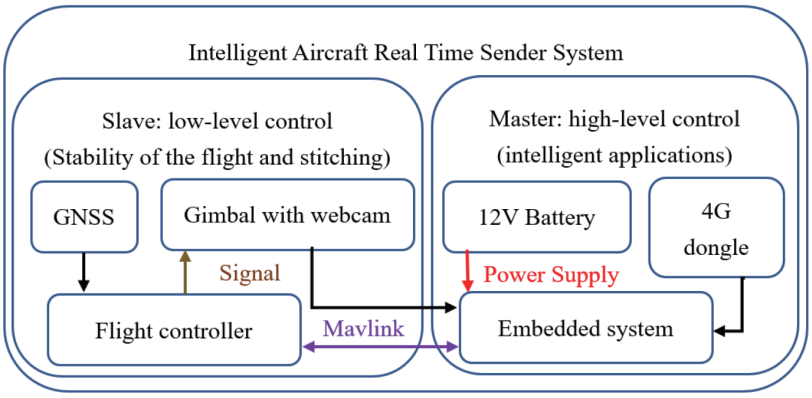


Fig. 2. (Color online) Architecture of the UAV sender.

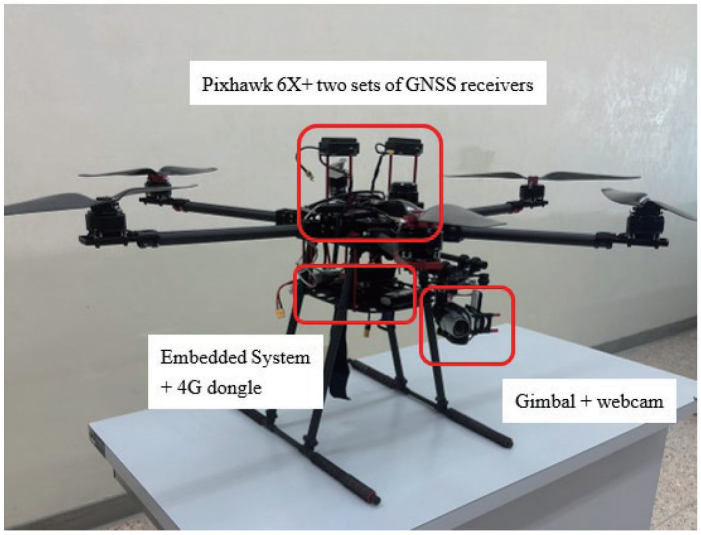


Fig. 3. (Color online) Hexacopter.

interference when there are devices with the same frequency. According to Ref. 20, instructions are set to change the Net ID to make it one-to-one. The Hexacopter_1 Telemetry Radio Net ID is 25, and the Hexacopter_2 Telemetry Radio Net ID is 196.

3. Image Stitching

Here, the feature-based image stitching algorithm is briefly introduced. First, feature detection and description are performed on the image, and then feature point matching is performed on two images with overlapping areas. Finally, the homography matrix is calculated and reprojected to obtain the image stitching result.

3.1 Feature detection and description

In this subsection, we introduce the feature detection and description methods that are used in this study: SIFT, ORB, and AKAZE.

3.1.1 SIFT

The advantage of SIFT is that it is invariant to image transformation, scaling, and rotation, and it is partially invariant to illumination changes, 3D rotation, and nonlinear geometric distortion. Scale space is used in an attempt to simulate the concept and method of human eyes observing objects in computer vision, that is, the degree of blur of the image. When the sigma is larger, the image is blurrier. Laplacian of Gaussian (LoG) can construct the scale space, but the calculation load is relatively high. In SIFT, the Difference of Gaussian (DoG) is used. Equation (1) is the Gaussian function, L in Eq. (2) is the Gaussian blur function, and the original image results from a convolution. Equation (3) is the DoG. Figure 4(a) shows its architecture and Fig. 4(b) shows the local extremum on the scale found from the DoG results; a pixel in the image is shown as X. Its eight neighboring pixels and nine pixels in the following scale are compared with nine pixels in the previous scale. If it is a local extremum, then it is a potential key point.

$$G(x,y,\sigma)=\frac{1}{2\pi\sigma^2}e^{-\left(x^2+y^2\right)/2\sigma^2} \tag{1}$$

$$L(x,y,\sigma)=G(x,y,\sigma)*I(x,y) \tag{2}$$

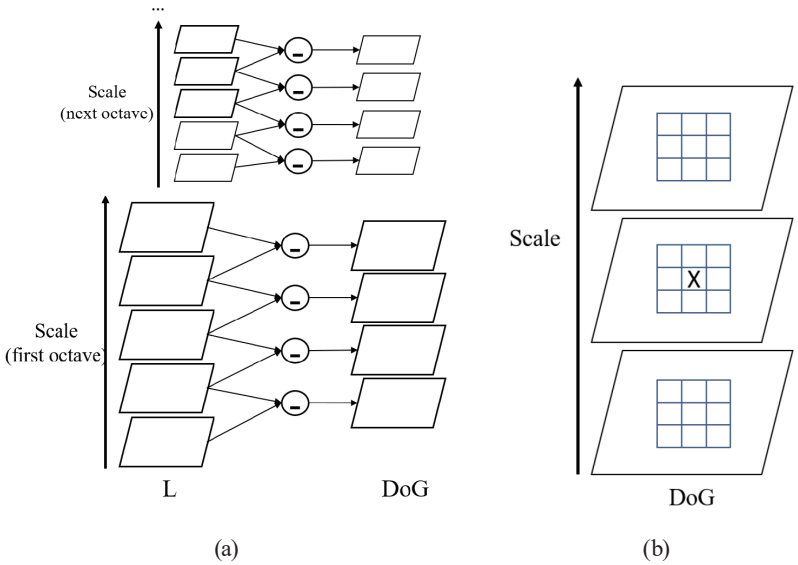


Fig. 4. (Color online) (a) Architecture of DoG and (b) results of DoG, modified from Ref. 4.

$$\begin{aligned} D(x,y,\sigma) &= \big(G(x,y,k\sigma) - G(x,y,\sigma)\big) * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma) \end{aligned} \tag{3}$$

After finding the potential key points, we use the Taylor series expansion filter on $D(x, y, \sigma)$ to obtain the exact extremum position, where D is calculated at the sample point, and $x = (x, y, \sigma)^T$ is the offset, derived in Eq. (4). The extreme value result is Eq. (5), and Eq. (6) can be obtained by bringing Eq. (5) back to Eq. (4) and then comparing $|D(\hat{x})|$ with the threshold value for screening.

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x \tag{4}$$

$$\hat{x} = -\frac{\partial^2 D}{\partial x^2}^{-1} \frac{\partial D^T}{\partial x} \tag{5}$$

$$D(\hat{x}) = D + \frac{1}{2} \frac{\partial D^T}{\partial x} \hat{x} \tag{6}$$

For stability, it is not enough to remove low-contrast key points. DoG responds strongly to the edge, so we use a 2 × 2 Hessian matrix (H) to calculate the principal curvature, as in Eq. (7). We can directly calculate the ratio as in Eqs. (8) and (9), let $\alpha = \gamma\beta$, bring it into Eq. (10), and finally filter low-contrast key points through Eq. (11).

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{7}$$

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \tag{8}$$

$$Det(H) = D_{xx} D_{yy} - \left(D_{xy}\right)^2 = \alpha\beta \tag{9}$$

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(\gamma\beta+\beta)^2}{\gamma\beta^2} = \frac{(\gamma+1)^2}{\gamma} \tag{10}$$

$$\frac{Tr(H)^2}{Det(H)} < \frac{(\gamma+1)^2}{\gamma} \tag{11}$$

We assign an orientation to each key point to achieve image rotation invariance. A Gaussian-smoothed image L with the closest scale, gradient magnitude $m(x, y)$ as in Eq. (12), and direction $\theta(x, y)$ as in Eq. (13), is chosen, computed in terms of pixel difference, and the overlay orientation histogram of 36 bins for 360 degrees is obtained as follows.

$$m(x,y)=\sqrt{\left(L(x+1,y)-L(x-1,y)\right)^2+\left(L(x,y+1)-L(x,y-1)\right)^2} \tag{12}$$

$$\theta(x,y)=\tan^{-1}\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right) \tag{13}$$

After creating the key point descriptor, the surrounding 16 × 16 neighborhood centered on the key point is selected and divided into sub-blocks of size 4 × 4. For each sub-block, we create eight bin-oriented histograms. Thus, there will be 128 bin values, as shown in Fig. 5.

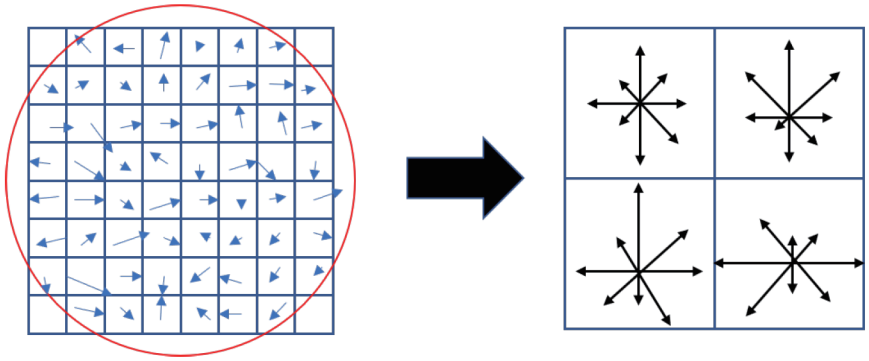
3.1.2 Oriented FAST and rotated BRIEF (ORB)

ORB uses the FAST feature descriptor and BRIEF feature description. There are four main contributions.[5]

1) Added fast and accurate orientation components based on FAST
2) Efficient computation of oriented BRIEF
3) Brief features of oriented variance and correlation analysis
4) A learning method for disassociating BRIEF under rotation invariance, which can achieve better performance in nearest-neighbor (NN) applications

The following is divided into three subsections for the introduction.

To judge a feature point, we compare the grayscale values around the candidate pixel point p. With p as the center, we define its intensity as Ip, select an appropriate threshold t, and compare 16 pixels on a circle with a radius of 3 pixels, as shown in Fig. 6. If the absolute difference



(a) (b)

Fig. 5. (Color online) Key point descriptor, modified from Ref. 4. (a) Image gradients. (b) Key point descriptor.

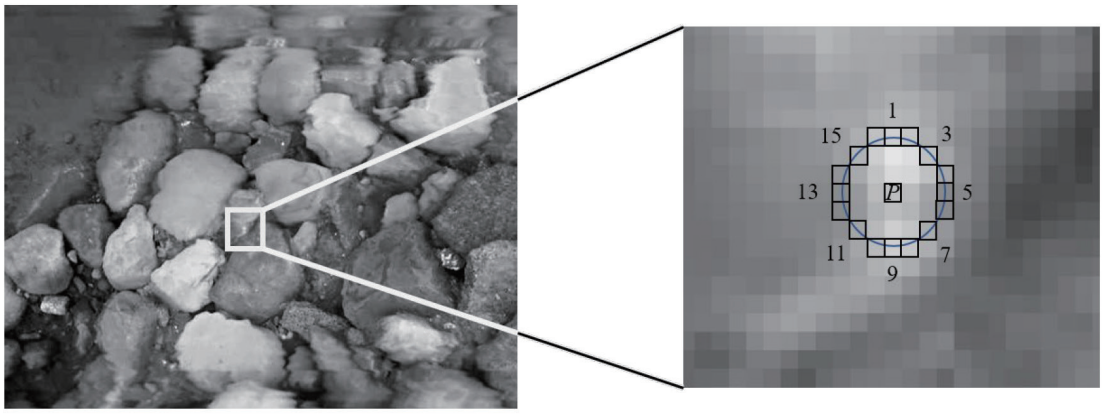


Fig. 6. (Color online) Image sampling example.

between the grayscale values of N continuous pixel points and p is greater than or equal to t, it is judged that the detection point is a feature point. First, to speed up the decision-making process, we calculate the four pixel points with serial numbers 1, 9, 5, and 13, and then judge other pixel points if at least three of them meet the conditions; otherwise, we discard point p.

Using the intensity centroid and assuming that the intensity of the corner is off-center, we estimate the direction using this vector, define the moments of a patch as Eq. (14), find the centroid Eq. (15) by calculating Eq. (14), and construct a patch from the center of the angle vector; the direction vector connecting the center o to the centroid c is determined using Eq. (16).

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{14}$$

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{15}$$

$$\theta = a\tan 2\left(m_{01}, m_{10}\right) \tag{16}$$

The BRIEF is used to create a binary feature descriptor with key points. Considering a Gaussian-smoothed image, a smoothed image patch is P, the binary test equation is Eq. (17), and $p(x)$ is the intensity of p at point x. Equation (18) is the n-bit binary string.

$$\tau(p;x,y) := \begin{cases} 1 & : p(x) < p(y) \\ 0 & : p(x) \geq p(y) \end{cases} \tag{17}$$

$$f_n(p) := \sum_{1 \leq i \leq n} 2^{i-1} \tau(p;x_i,y_i) \tag{18}$$

A more efficient method is to define a 2 × n matrix, such as Eq. (19), for any function set (x_i, y_i) of n binary tests of key points in accordance with the key points and use the θ and rotation matrix R_θ obtained by FAST to calculate the rotational coordinates as in Eq. (20).

$$S = \begin{pmatrix} x_1, \ldots, x_n \\ y_1, \ldots, y_n \end{pmatrix} \tag{19}$$

$$S_\theta = R_\theta S \tag{20}$$

3.1.2 AKAZE

AKAZE is an accelerated version of KAZE, which uses fast explicit diversion (FED) to speed up the co-construction of pyramid spaces with nonlinear scales. KAZE uses nonlinear diffusion filtering to construct the scale space, described by a nonlinear partial differential equation [Eq. (21)]. Equation (22) is the conductance function, ∇L_σ is the gradient of the original image after Gaussian smoothing, and the function g is defined in Eq. (23).

$$\frac{\partial L}{\partial T} = \mathrm{div}\left(c\left(x, y, t\right) \cdot \nabla L\right) \tag{21}$$

$$c\left(x, y, t\right) = g\left(\left|\nabla L_\sigma\left(x, y, t\right)\right|\right) \tag{22}$$

$$g = \frac{1}{1 + \frac{\left|\nabla L_\sigma\right|^2}{\lambda^2}} \tag{23}$$

Iterative box filters approximate Gaussian kernels with good quality and are easy to implement. The primary step is to execute n explicit diffusion steps for M cycles, whose step size τ_j is derived from the decomposition of the box filter as in Eq. (24). The stop time θ_n corresponding to one FED cycle is given by Eq. (25). The FED cycle has n variable steps τ_j resulting in Eq. (26).

$$\tau_j = \frac{\tau_{max}}{2\cos^2\left(\pi \frac{2j+1}{4n+2}\right)} \tag{24}$$

$$\theta_n = \sum_{j=0}^{n-1} \tau_j = \tau_{max} \frac{n^2 + n}{3} \tag{25}$$

$$L^{i+1,j+1} = \left(I + \tau_j A\left(L^i\right)\right) L^{i+1,j},\ j = 0,...,n-1 \tag{26}$$

We construct the scale space on the basis of the evolution time, use a diffusion function such as Eq. (27), and convert the zoom level of the discrete set in pixel units into a time unit, such as Eq. (28).

$$\begin{gathered} \sigma_i(o,s) = 2^{o+\frac{s}{S}}, \\ o \in [0 \ldots O-1], s \in [0 \ldots S-1], i \in [0 \ldots M] \end{gathered} \tag{27}$$

$$t_i = \frac{1}{2}\sigma_i^2, i \in [0 \ldots M] \tag{28}$$

The feature points are extracted using the local maximum of the normalized Hessian matrix of different scales:

$$L^i_{Hessian} = \left(\frac{\sigma_i}{2^{o^i}}\right)^2 \left(L^i_{xx}L^i_{yy} - L^i_{xy}L^i_{xy}\right). \tag{29}$$

3.2 Feature point matching

In this subsection, we introduce three different feature point matching methods that are used in this study: brute-force matching, FLANN-based matching, and K-NN matching.[21,22]

(a) Brute-force matching

We select a feature descriptor in the first image and each feature descriptor in the second image to calculate the distance. The shortest distance is the matching point, and we repeat the selection until all the feature descriptors in the first image are calculated.

(b) FLANN-based matching

FLANN-based matching uses a set of algorithms optimized for fast NN searches (NSSs) and high-dimensional features in large datasets. It works faster than brute-force matching for large datasets. The NSS problem is defined as follows: given a set of points $P = (p_1, p_2, ..., p_n)$ in the metric space X and a new query point $q \in X$, we find the closest point in P to q.

(c) K-NN matching

The algorithm finds the two points closest to each feature point in the registration image from each feature point in the two images. We calculate the ratio of the NN distance to the second NN distance, compare the threshold with the ratio, and when the ratio is less than the threshold, select the feature point corresponding to the closest neighbor distance as the best matching point.[23,24]

3.3 Homography matrix and eliminate mismatch points

The homography matrix H is shown in Eq. (30). x, y are the coordinates, which means that the x_i, y_i of the right image will be converted to the x_i', y_i' of the left image through the H matrix during stitching.[25,26]

$$s_i \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{30}$$

We find the homography in OpenCV by which one of various methods of eliminating mismatch points can be chosen in accordance with the need. In this study, random sample consensus and progressive sample consensus are used for comparison.[27–29]

(a) RANSAC

The advantages of RANSAC are its simple structure, easy implementation, and strong robustness, and if it widely used to eliminate mismatching. However, owing to its random sampling, it can only sometimes find the best feature correspondences and does not perform well with many false matches. The procedural steps of RANSAC are as follows.

(1) Randomly select four sample data points from the well-matching feature points and calculate the homography matrix H.
(2) Calculate the projection error between all the data in the dataset and the matrix H. If the error exceeds the threshold, add the inner point set X.
(3) If the elements of the current interior point set X are better than the optimal interior point set $I_$best, update $I_$best = X and update the number of iterations k.
(4) Repeat steps (1) to (3) k times and obtain the best H.

(b) PROSAC

The PROSAC algorithm is used to improve the randomness of the RANSAC algorithm. It is superior to RANSAC in terms of robustness and computational efficiency. The algorithm first sorts the data from high to low and only draws samples from high-quality data. Repeated verification was performed, and the best solution was obtained.

3.4 Real-time image stitching test

We first present the experimental results of real-time image stitching using an octocopter and a hexacopter, as shown in Fig. 7. SIFT is used for feature detection and description. Brute-force matching with K-NN matching is used for feature point matching, and the homography matrix is found using RANSAC to eliminate mismatches. Figures 8 and 9 are the image stitching results with different degrees of overlap. Table 1 presents the number of feature points and the number of matched feature points determined from Figs. 8 and 9.

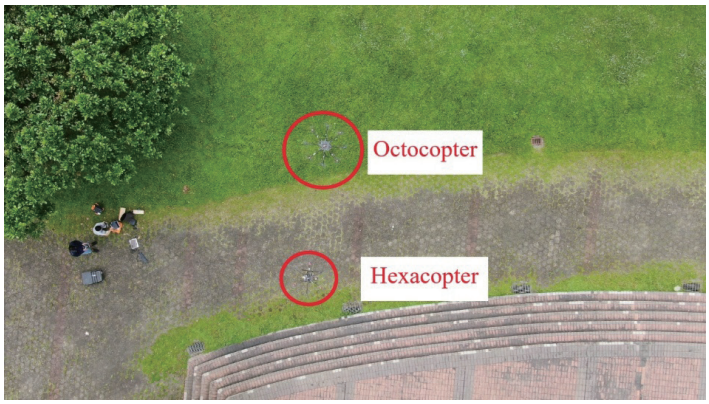
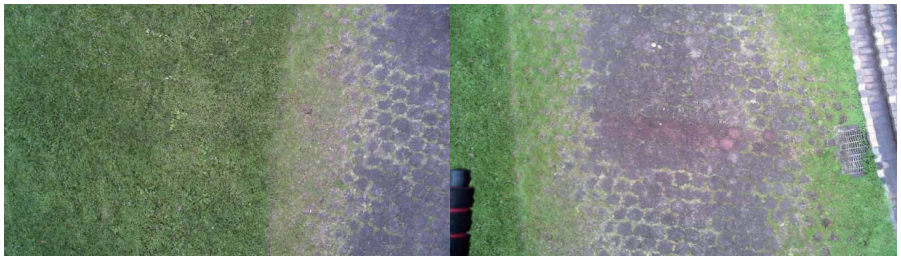
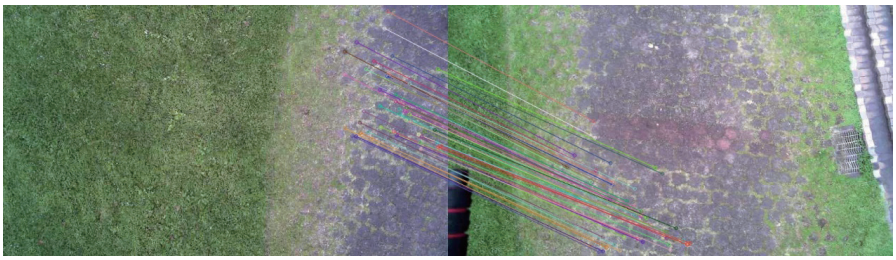


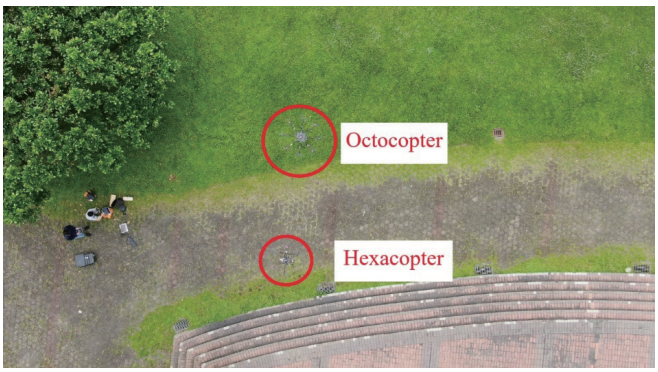
Fig. 7. (Color online) UAVs used in the experiment.



(a) (b)

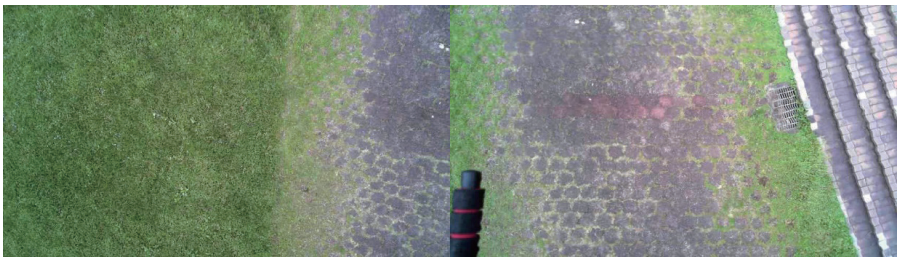


(c)

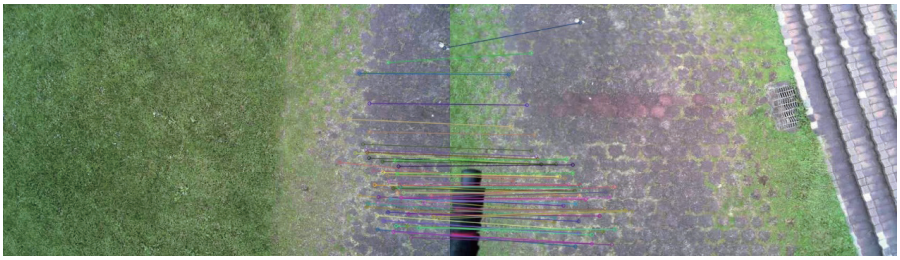


(d)

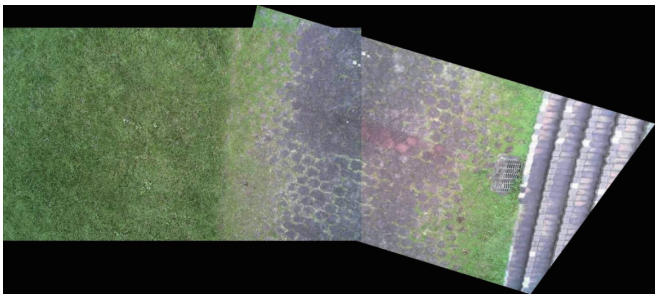
Fig. 8. (Color online) First group of image stitching. (a) Octocopter viewing image. (b) Hexacopter viewing image. (c) Feature point matching result. (d) Image stitching results.



(a) (b)



(c)



(d)

Fig. 9. (Color online) Second group of image stitching. (a) Octocopter viewing image. (b) Hexacopter viewing image. (c) Feature point matching result. (d) Image stitching results

Table 1
Image stitching feature point data.

Group	Feature points of (a)	Feature points of (b)	Matched feature points
Group 1	6027	5389	56
Group 2	5605	5072	64

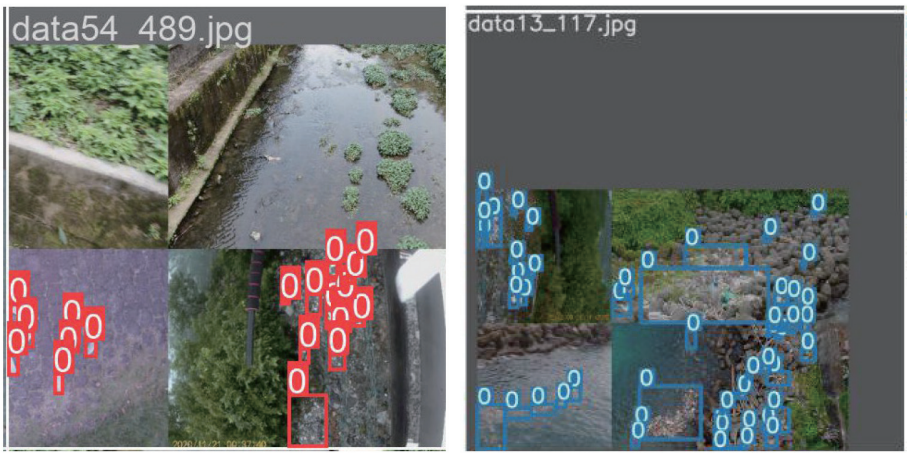
4. Debris Pattern Recognition

YOLOv5 is mainly divided into five models, from small to large, namely, YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x, and can be trained with different hyperparameters. When YOLOv5 and YOLOv7 input training datasets, they will use adaptive image scaling to make the image sizes consistent and use data augmentation techniques and adaptive anchor box calculation processing to obtain more diverse training datasets. In YOLOv5 and YOLOv7, different data augmentation techniques are used to augment the training dataset.

Methods include mosaic augmentation, copy-paste augmentation, random affine transformation, hybrid augmentation, album, HSV augmentation, and random horizontal flip. A commonly used method is mosaic enhancement, which combines four training images into one. Random affine transformation involves randomly rotating, scaling, translating, and cropping four training images into one image. MixUp augmentation involves combining two images and their labels. HSV augmentation involves changing the image hue and color saturation. Random horizontal flip denotes randomly flipping the image horizontally. Copy-paste augmentation in YOLOv5 is described as an innovative data enhancement method; the patch is copied from the image and pasted onto a random image to generate new training samples.[30] Figure 10(a) shows the YOLOv5 training dataset after data augmentation. Figure 10(b) shows the YOLOv7 training dataset after data augmentation. The boxes and numbers 0 in Fig. 10 are the categories labeled for the training set.

In the training dataset, there is a problem with the different sizes of pictures. Through adaptive picture scaling, the sizes are scaled to be uniform. In the input training network, the scaling ratio is first calculated using the original length and width and the scaled length and width. For a small ratio, we calculate the scaled size and the value for filling the gray edge. In YOLOv5, the minimum gray edge is filled after calculation, reducing unnecessary information that affects the training reasoning. In the network training, the network outputs the prediction frame on the basis of the initial anchor frame, compares it with the actual frame, calculates the difference between the two, and then updates and iterates the network parameters. YOLOv5 and YOLOv7 embed this function into the program code and determine the best anchor frame values in different training sets through K-means at each training time. In Ref. 10, real-time object detectors are compared among YOLOv7, YOLOR, PPYOLOE, YOLOX, Scaled-YOLOv4, and YOLOv5. YOLOv7 is fast and also has average precision. YOLOv7 mainly has six models and three hyperparameters, which can be trained in different combinations.

By scaling the designed model to generate models of different sizes, we can further apply it to other computing devices. Model scaling methods usually use various scaling factors, such as



(a) (b)

Fig. 10. (Color online) (a) YOLOv5 and (b) YOLOv7 training datasets after data augmentation.

resolution (the size of the input image), depth (the number of layers of the network), and width (the number of channels of the feature map), to achieve a good trade-off in the amount of computation, inference speed, and accuracy during training. In the scaling of the series model (such as VoVNet), if the depth is enlarged or reduced, the input width of the subsequent transition layer will increase or decrease accordingly, leading to a poor training effect and complicated calculation. The compound model scaling method only needs to scale the depth in the computational block and the width of the rest of the transition accordingly to maintain the properties of the model at the design time and maintain the best structure. Model reparameterization technology can be regarded as an ensemble technology that is mainly divided into two categories, namely, model level and module level. For model-level reparameterization, there are mainly two methods. One is to train the same model with different training data and then average the weights of multiple training models. The other is to determine the weights of models with various iterations. The concept of module-level reparameterization is to split a module into multiple identical or different module branches during training and then integrate numerous branch modules into completely equivalent modules during inference. However, not all proposed reparameterization modules can be applied to different architectures, and their inappropriateness will affect the accuracy.

In this study, we show the training results of YOLOv7 using the PyTorch framework. It mainly uses five model architectures with three hyperparameters, but the parameters cannot be adjusted to be sufficiently consistent to enable detailed comparison owing to computer limitations. From the eight training results, Objectness and mAP@.5 are mainly analyzed. Objectness does not converge very smoothly. After training for 1000 epochs, mAP@.5 has a downward or oscillating trend, so we can reduce the number of epochs required for training. However, Objectness is oscillating; in this case, we can increase the number of epochs to confirm whether there is a better training result, and then confirm whether there is overfitting. In Table 2, the overall performance can be indicated by mAP@.5. In the training results, the highest value of mAP@.5 is obtained with the model YOLOv7 with the hyperparameter P5, and the lowest with the model YOLOv7 with the hyperparameter Tiny.

Table 2
Training results for various models.

Model	Hyperparameter	Image size	Batch size	Precision	Recall	mAP@.5
YOLOv7-tiny	P5	640*640	32	0.8127	0.7414	0.8057
	Tiny			0.8023	0.7588	0.7763
YOLOv7	P5			0.82	0.7846	0.819
	Tiny			0.7997	0.7741	0.7531
YOLOv7-x	P5		24	0.8155	0.7703	0.8182
	Tiny			0.8106	0.7778	0.7751
YOLOv7-w6	P6	1280*1280	12	0.8341	0.7572	0.8065
YOLOv7-d6	P6		8	0.828	0.736	0.797

5. Experiments and Results

In this research, we used NVIDIA's embedded system and the UAV Riverine Waste Monitoring System for real-time debris recognition and real-time scene image stitching.

5.1 Real-time image stitching

Figure 11 shows the process of real-time image stitching in this study; our program is a modified version of that in Ref. 31.

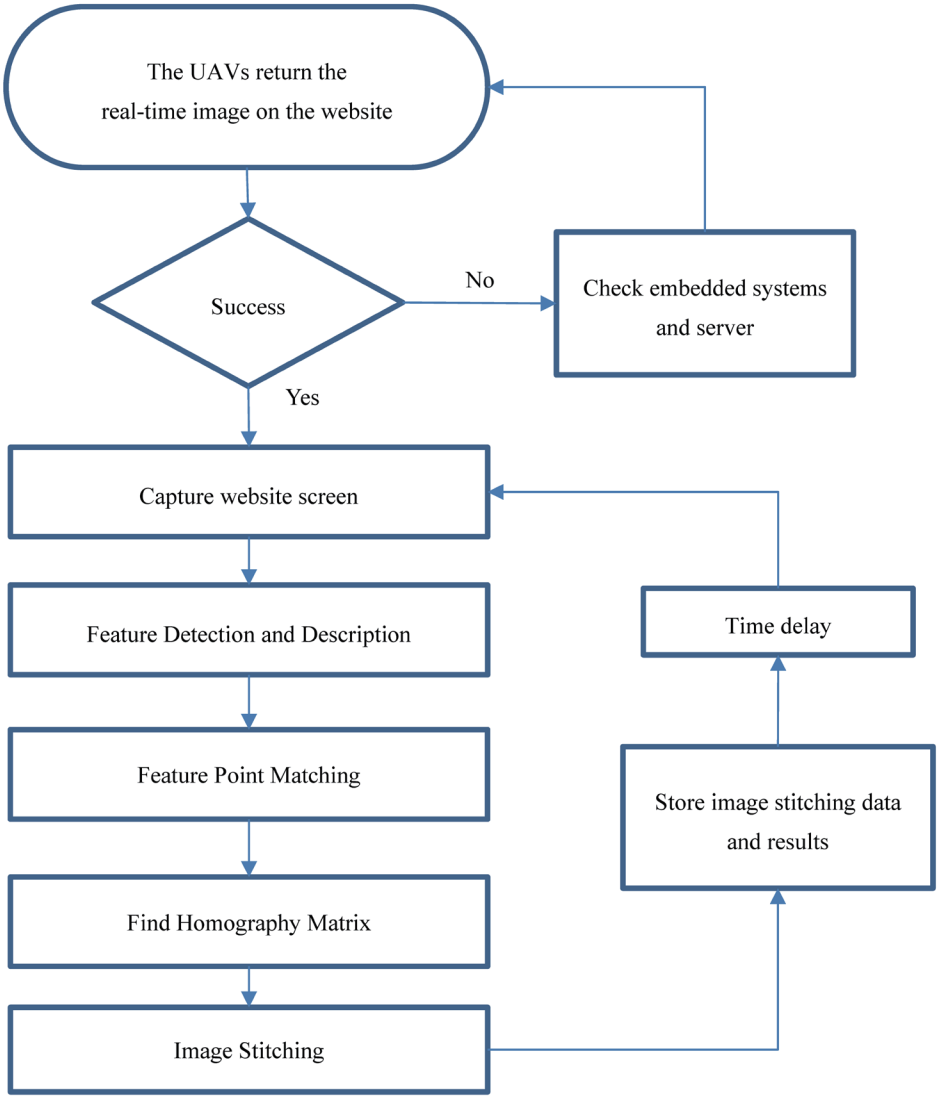


Fig. 11. (Color online) Real-time image-stitching process.

5.2 Image stitching

Next, the real-time image stitching of different scenes is introduced. Figure 12 shows the two UAVs used in this experiment.

(a) Feature detection and description: SIFT

Figure 13 shows apparent color differences between images (a) and (b); the dislocation of stones is more prominent in (d). When the distance between two UAVs is too large, the image overlap rate will decrease, leading to image stitching failure.

Fig. 12. (Color online) Hexacopters used in the experiment.

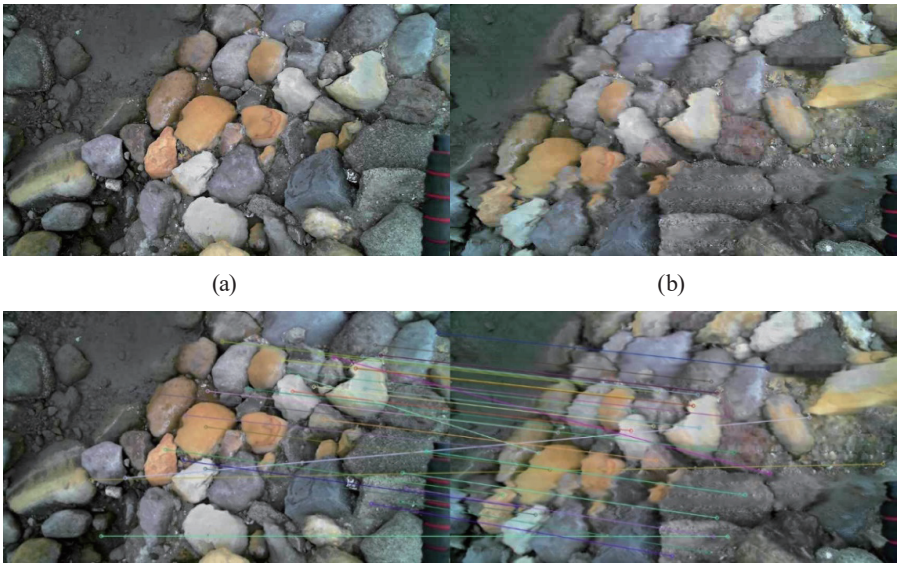
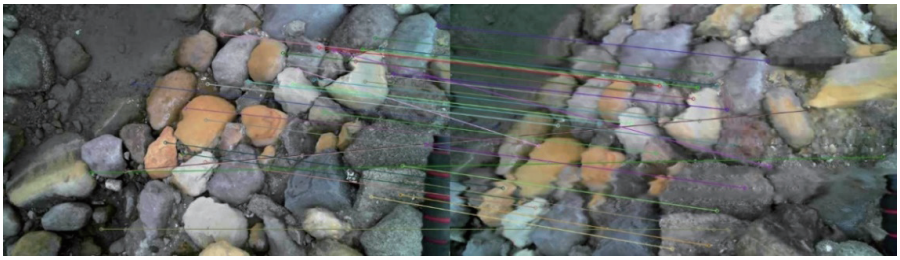


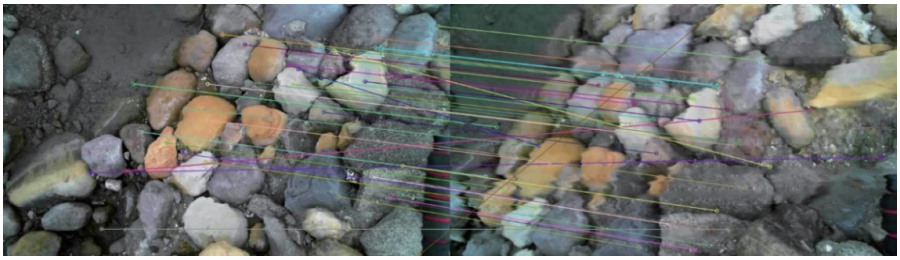
Fig. 13. (Color online) First group of image stitching. (a) Hexacopter_1 viewing image. (b) Hexacopter_2 viewing image. (c) Matching results_1 (brute-force matching + K-NN matching). (d) Image stitching results_1 (find homography matrix using RANSAC). (e) Matching results_2 (brute-force matching + K-NN matching). (f) Image stitching results_2 (find homography matrix using PROSAC). (g) Matching results_3 (FLANN-based matching + K-NN matching). (h) Image stitching results_3 (find homography matrix using RANSAC). (i) Matching results_4 (FLANN-based matching + K-NN matching). (j) Image stitching results_4 (find homography matrix use PROSAC).

(d)



(e)

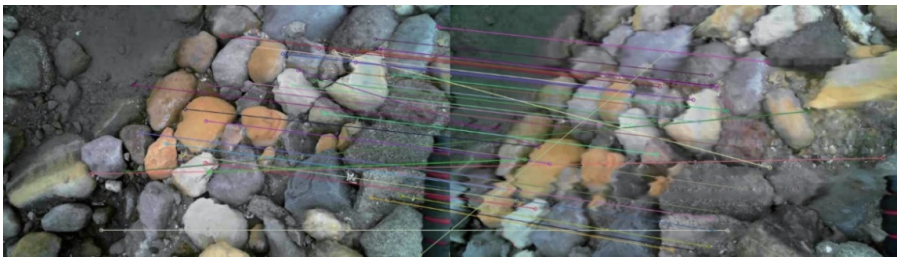
(f)



(g)

Fig. 13. (Color online) (Continued) First group of image stitching. (a) Hexacopter_1 viewing image. (b) Hexacopter_2 viewing image. (c) Matching results_1 (brute-force matching + K-NN matching). (d) Image stitching results_1 (find homography matrix using RANSAC). (e) Matching results_2 (brute-force matching + K-NN matching). (f) Image stitching results_2 (find homography matrix using PROSAC). (g) Matching results_3 (FLANN-based matching + K-NN matching). (h) Image stitching results_3 (find homography matrix using RANSAC). (i) Matching results_4 (FLANN-based matching + K-NN matching). (j) Image stitching results_4 (find homography matrix use PROSAC).

(h)



(i)

(j)

Fig. 13. (Color online) (Continued) First group of image stitching. (a) Hexacopter_1 viewing image. (b) Hexacopter_2 viewing image. (c) Matching results_1 (brute-force matching + K-NN matching). (d) Image stitching results_1 (find homography matrix using RANSAC). (e) Matching results_2 (brute-force matching + K-NN matching). (f) Image stitching results_2 (find homography matrix using PROSAC). (g) Matching results_3 (FLANN-based matching + K-NN matching). (h) Image stitching results_3 (find homography matrix using RANSAC). (i) Matching results_4 (FLANN-based matching + K-NN matching). (j) Image stitching results_4 (find homography matrix use PROSAC).

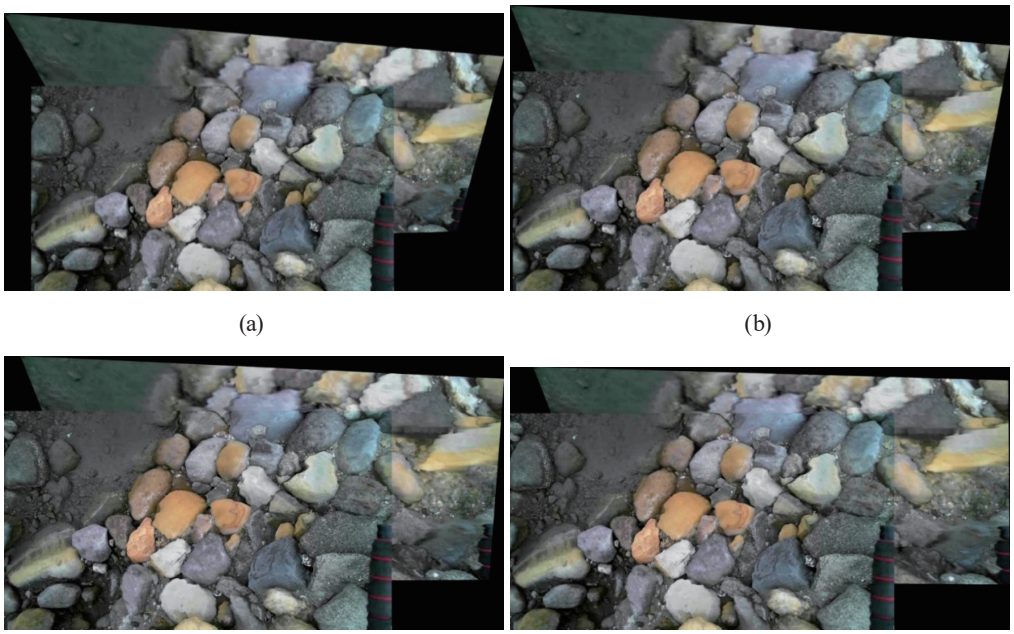
(b) Feature detection and description: AKAZE

Figure 14 shows four stitching results based on AKAZE.

(c) Feature detection and description: ORB

Figure 15 shows four stitching results based on ORB.

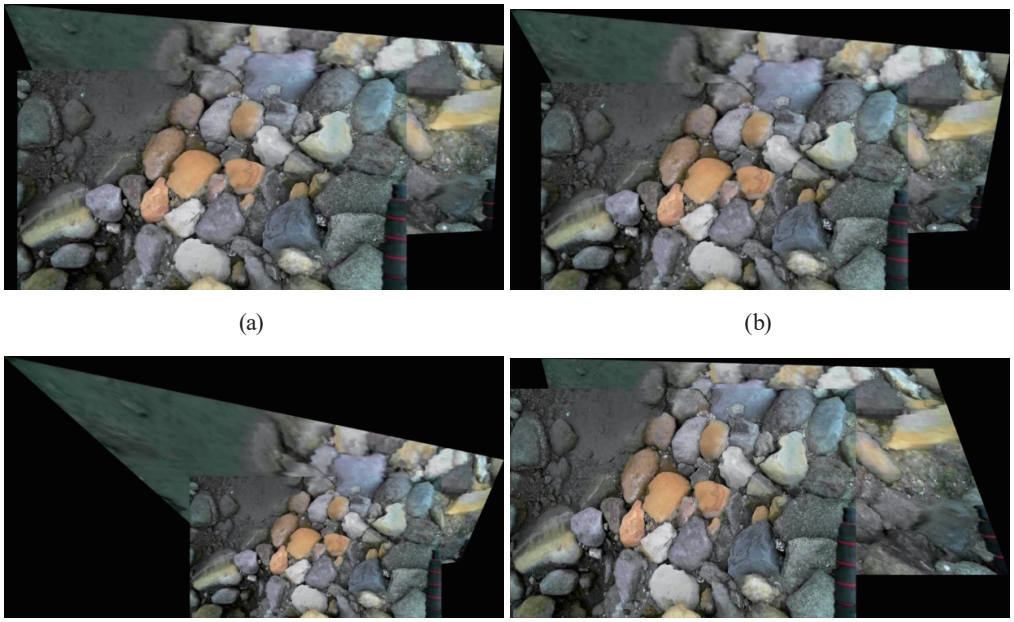
Table 3 shows that feature detection and description take a short time, but the number of feature points obtained is relatively large. In Table 4, Matching result_1 and Matching result_2 have the same number of feature point matches obtained when using brute-force matching.



(a) (b)

(c) (d)

Fig. 14. (Color online) Second group of image stitching. (a) Image stitching results_1. (b) Image stitching results_2. (c) Image stitching results_3. (d) Image stitching results_4.



(a) (b)

(c) (d)

Fig. 15. (Color online) Third group of image stitching. (a) Image stitching results_1. (b) Image stitching results_2. (c) Image stitching results_3. (d) Image stitching results_4.

Table 3
Feature detection and description time and number of feature points.

Method	Feature detection and description time (s)	Feature points in Fig. 13(a)	Feature points in Fig. 13(b)
SIFT	0.5625	3213	1879
AKAZE	0.375	1353	962
ORB	0.1875	2000	2000

Table 4
Number of feature point matches for different methods.

Method	Matching result_1	Matching result_2	Matching result_3	Matching result_4
SIFT	35	35	42	41
AKAZE	87	87	97	99
ORB	31	31	38	37

Because the distance is compared without complex calculation optimization in brute-force matching, the same effect is obtained in the same scene with the same number of feature points; the number of matched feature points is small because of multiple feature point matching screenings.

Feature detection and description are combined with different feature point matching methods in image stitching. Twelve combinations of methods are tested, as shown in Table 4. The performance from good to poor is SIFT, AKAZE, and ORB. The best stitched image is the one obtained using SIFT and FLANN-based matching + K-NN matching with RANSAC stitching. The computing time for feature detection and description increased in the order of ORB, AKAZE, and SIFT. Although SIFT takes more time, the total computing time in image stitching is still acceptable for real-time debris inspection.

5.3 Debris pattern recognition

Since the real-time UAV waste monitoring system uses YOLO of the Darknet framework and cannot use YOLO of the Pytorch framework, Darknet was used for training YOLOv5 and YOLOv7. The training results of YOLOv5-tiny, YOLOv7-tiny, YOLOv7, and YOLOv7x show that the average loss converges, and mAP is also stable at 76, 74, 74, and 75%, respectively. Therefore, the results show no need to increase the number of training epochs. Figure 16 shows the UAV waste monitoring system during a debris identification test: its coordinates and area size are sent back to the monitoring terminal, and then a heat map, shown in Fig. 17, is prepared. The identification results are shown in Fig. 18, including the polluted area, the longitude and latitude of the location, UAV information, and blue points indicating garbage. Figures 19–21 show the real-time debris recognition results at the Keelung River area, where the identification was carried out at a viewing angle of 45 degrees. Figures 19 and 21 show the river surface's reflection and the misidentification of stones. The proposed system can also identify small and inconspicuous debris and presents the highest confidence index of 66.77. Table 5 shows the correct rate and false rate of garbage identification, and Fig. 20 shows missing marks because of the relatively small size of garbage, resulting in a low correct rate.

Fig. 16. (Color online) NTOU garbage identification test.

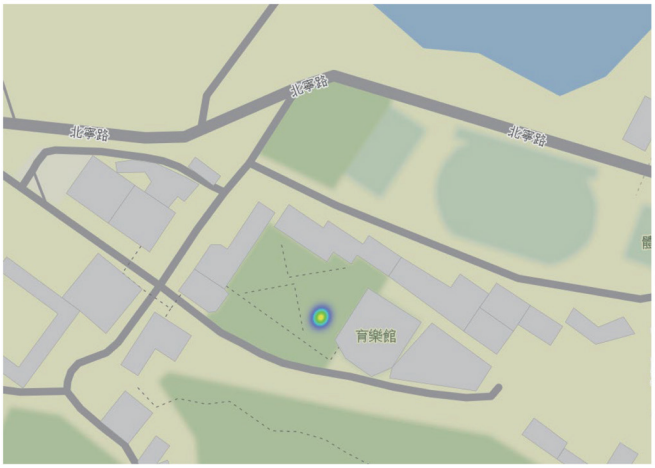


Fig. 17. (Color online) Monitoring terminal: garbage heat map.

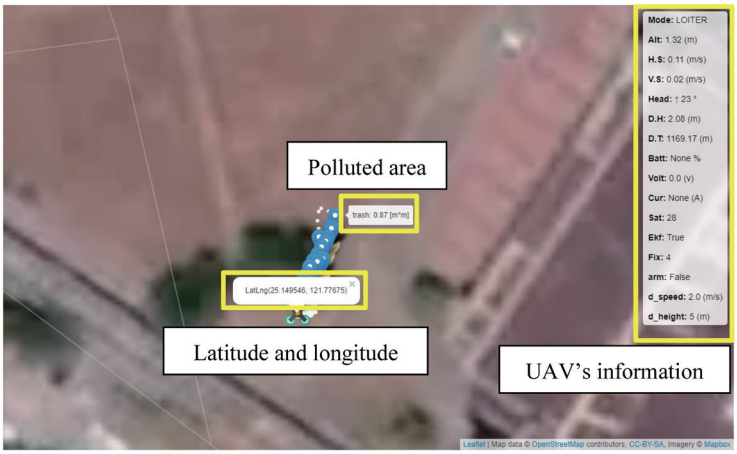


Fig. 18. (Color online) Monitoring terminal: map of garbage identification results.

Fig. 19. (Color online) Qixian Bridge-Keelung River real-time garbage identification results_1.

Fig. 20. (Color online) Qixian Bridge-Keelung River real-time garbage identification results_2.

Fig. 21. (Color online) Qixian Bridge-Keelung River real-time garbage identification results_3.

Table 5
Garbage identification result data.

	Fig. 19	Fig. 20	Fig. 21
Correct rate (%)	60	25	80
False rate (%)	40	75	20

When using the UAV waste monitoring system for real-time debris recognition, the data (area size, current coordinate location, debris type) are streamed through kafka, the real-time images are transmitted through imageZMQ, and the results are sent to the monitoring terminal; the flight height is about 2 m. The take-off location and the river bank have a height difference, so the accurate measurement is about 6 m. Owing to environmental factors, performing real-time debris recognition and scene image stitching simultaneously requires much work.

6. Conclusions

We used the real-time UAV waste monitoring system for real-time debris recognition and proposed a multiple-drone system with deep learning neural networks, YOLO, to quickly and effectively detect river banks and barriers and send their current coordinates, debris area size, and images to the ground control center. Feature detection and description were combined with different feature point matching methods in image stitching. The results of experiments showed that real-time image stitching can be accomplished in a short time, which enables real-time debris recognition for practical implementation. The proposed UAV waste monitoring system can generate a heat map of debris and send its coordinates and area size back to the ground station for further environmental monitoring usage. However, owing to environmental factors, real-time image stitching can only be achieved in certain ways. The experiments and result analyse revealed that the main factors affecting the stitching results are as follows. (1) The photography equipment used is autofocus. When the UAV cannot be stabilized at a high altitude, it will lead to an inability to accurate focusing. (2) Vibration is caused by the flying UAV. (3) When the wind speed is high, the UAV cannot maintain its fixed direction angles (yaw) or fixed direction. To solve the above problems, dual GNSS with RTK technology and a better flight controller were used, but there was still much noise in the real-time returned image, which led to poor stitching results.

The real-time debris recognition experiment was carried out at the Keelung River. From the results, it can be confirmed that even small debris in the scene was recognized. Still, there also were misidentifications, such as recognizing stones and reflection of the river surface as garbage. Some of the confidence indexes presented were low, only 15–20%. In the future, the real-time UAV waste monitoring system can be modified to use YOLO of the Pytorch framework for debris recognition or other methods to achieve UAV real-time debris recognition. Further increasing the number of training sets of different types of debris around the riverbank can improve the identification accuracy and diversity. In real-time image stitching, the noise on UAV-captured images should be reduced, and topnotch photographic equipment should be used to perform high-quality image stitching.

References

1 Y. H. Liao and J. G. Juang: Proc. 2021 IEEE Int. Conf. System Science and Engineering (2021) J210079.
2 National Taiwan Ocean University: https://github.com/LiaoSteve/HAIDA-Trash-Dataset-High-Resolution-Aerial-image (accessed July 2021).
3 D. Kumareswaran, N. Nasir, M. Z. A. Rahman, and A. S. M. Supa'at: Proc. 2021 Int. Conf. Communication, Control and Information Sciences (2021) 1–6.
4 D. G. Lowe: Int. J. Comput. Vision **60** (2004) 91. https://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94
5 E. Rublee, V. Rabaud, K. Konolige, and G. Bradski: Proc. 2011 Int. Conf. Computer Vision (2011) 2564–2571.
6 P. Alcantarilla, J. Nuevo, and A. Bartoli: Proc. 2013 British Machine Vision Conf. (2013) 1–11.
7 S. A. K. Tareen and Z. Saleem: Proc. 2018 Int. Conf. Computing, Mathematics and Engineering Technologies (2018) 1–10.
8 Q. Yan, Q. Li, and T. Zhang: Proc. 2020 IEEE 5th Int. Conf. Signal and Image Processing (2020) 524–529.
9 J. Liu, Q. Wei, and Y. Bai: Proc. 2021 IEEE 3rd Int. Conf. Civil Aviation Safety and Information Technology (2021) 1310–1313.
10 C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao: Proc. 2023 IEEE/CVF Conf. Computer Vision and Pattern Recognition (2023) 7464–7475.
11 S. Wang: Master Theses, UAV Positioning and Orientation Using Low-Cost Dual-Frequency GNSS RTK Technology (National Taiwan Ocean University, Taiwan, 2020).
12 Microsoft: https://answers.microsoft.com/en-us/windows/forum/all/webcam-microsoft-lifecam-studio-1080-hd-software/4deafd26-d329-4806-bb40-a1f2f2e8b523 (accessed March 2023).
13 J. Bass, imageZMQ: https://doi.org/10.5281/zenodo.12770292 (accessed May 2023).
14 NVIDIA: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-nx/ (accessed May 2023).
15 Seedstudio: https://www.seeedstudio.com/Jetson-20-1-H2-p-5329.html?queryID=e9b64ee955bbe6da0e454ff399dbf468&objectID=5329&indexName=bazaar_retailer_products (accessed May 2023).
16 Holybro: https://docs.holybro.com/autopilot/pixhawk-6x/overview (accessed May 2023).
17 ArduPilot: https://ardupilot.org/copter/ docs/common-rtk-correction.html#common-rtk-correction (accessed July 2022).
18 ArduPilot: https://ardupilot.org/copter/docs/common-gps-for-yaw.html#common-gps-for-yaw (accessed July 2022).
19 Holybro: https://docs.holybro.com/telemetry-radio/sik-telemetry-radio-v3 (accessed April 2023).
20 ArduPilot: https://ardupilot.org/copter/docs/common-configuring-a-telemetry-radio-using-mission-planner.html (accessed April 2023).
21 OpenCV: https://docs.opencv.org/3.4/dc/dc3/tutorial_py_matcher.html (accessed January 2023).
22 Github: https://github.com/flann-lib/flann (accessed February 2023).
23 Y. Wang, X. Yang, X. Wang, C. Ke, and Q. Wang: Proc. 2020 Int. Conf. Computer Vision, Image and Deep Learning, (2020) 536–541.
24 N. Pudchuen and C. Deelertpaiboon: Proc. 2018 Int. Electrical Engineering Congress (2018) 1–4.
25 OpenCV: https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html#projective_transformations (accessed June 2023).
26 C. W. Shu and X. Z. Xiao: Proc. 2018 IEEE Int. Conf. Progress in Informatics and Computing (2018) 246–249.
27 H. K. Jeon, J. M. Jeong, and K. Y. Lee: Proc. 2015 Int. SoC Design Conf. (2015) 91–92.
28 S. Wang, Y. Zhang, W. Wang, Y. Zhao, and S. Zhu: Proc. 2018 IEEE Int. Conf. Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data (2018) 1707–1713.
29 S. A. Bakar, X. Jiang, X. Gui, G. Li, and Z. Li: J. Phys.: Conf. Ser. **1624** (2020) 042023. https://iopscience.iop.org/article/10.1088/1742-6596/1624/4/042023
30 J. Terven, D. M. Córdova-Esparza, and J. A. Romero-González: Mach. Learn. Knowl. Extr. **5** (2023) 1680. https://doi.org/10.3390/make5040083
31 Github: https://github.com/sachin-vs/UAV-Image-stitching (accessed January 2023).