ACCSCIENCE PUBLISHING

An International Journal of Optimization and Control: Theories & Applications
ISSN: 2146-0957 eISSN: 2146-5703
Vol.16, No.5, pp.1643-1667 (2026)
https://doi.org/10.36922/IJOCTA026170066

IJOCTA

RESEARCH ARTICLE

Optimization of aerial image stitching for garbage identification

Cheng-Chien Su[1] and Jih-Gau Juang[2]*

[1]Department of Hardware R&D, Wincomm Corporation, Hsinchu, Taiwan

[2]Department of Communications, Navigation and Control Engineering, Faculty of Electrical Engineering and Computer Science, National Taiwan Ocean University, Keelung, Taiwan
jackies@wincomm.com.tw; jgjuang@mail.ntou.edu.tw

ARTICLE INFO

Article history:
Received: April 23, 2026
Revised: June 22, 2026
Accepted: June 23, 2026
Published Online: July 8, 2026

Keywords:
Unmanned aerial vehicles
Image identification
Dynamic image stitching
Deep learning
Stitch fusion

ABSTRACT

To improve the environmental quality and protect local ecosystems, riverine waste monitoring systems are needed to identify waste along riverbanks. Since most riverbank areas are wide, a single unmanned aerial vehicle (UAV) camera cannot capture the entire area. Hence, the use of multiple UAVs and image stitching for river scene reconstruction is required. This study applies image processing with multiple UAVs to identify garbage. First, image stitching employed the scale-invariant feature transform, Oriented Features from Accelerated Segment Test and Rotated Binary Robust Independent Elementary Features, and accelerated KAZE for feature detection and description. Real-time scene stitching was performed using feature point matching and filtering, and seam removal was achieved with an averaging-weighted mask. Two image-stitching datasets were tested, and the proposed method preserves 100% of the content of the original images. The computation time was reduced by 60% compared to the conventional method. Secondly, the optimized stitching results were used to further identify marine pollution using the You Only Look Once (YOLO) model. Different types of YOLO networks were used to identify waste pollution. An optimal YOLO model that identified waste items with 100% recognition and accuracy was obtained. The current coordinates, waste types, area size, and real-time images of the waste pollution were sent to the monitoring terminal. This enabled the analysis of waste distribution and facilitated attempts to trace the source of the waste by comparing training results and the identification effectiveness of different models.

1. Introduction

Many urban rivers pass through densely populated areas, accumulating waste along riverbanks and embankments. Carried downstream by river flow, this waste contaminates water and harms aquatic environments and local ecosystems. Given the limitations of government resources, we propose using unmanned aerial vehicles (UAVs) to detect and monitor waste along river corridors. By capturing and stitching images of affected areas, UAVs can help identify uncollected debris, enabling government agencies to manage riverine waste more effectively and mitigate pollution.

Cities are often crossed by rivers of varying sizes, with numerous bridges, banks, embankments, and piers. Despite existing laws and regulations, littering remains a persistent challenge. To improve the environmental quality and protect local ecosystems, we previously introduced a real-time UAV-based riverine waste monitoring system[1] designed to identify waste along riverbanks. In the present study, we extend the system to incorporate multiple UAVs and to stitch multiple images of river scenes. By providing government officials with clear, accurate information on the location and extent of riverine waste, the enhanced system supports

*Corresponding Author

more efficient waste management and removal efforts.

In a previous study, we developed an innovative real-time UAV-based riverine waste monitoring system that leverages the You Only Look Once (YOLO) object detection framework.[1] We also created the HAIDA garbage dataset,[2] which provides the training data necessary for the YOLO model. The system can be applied across various scenarios, using a website to provide real-time images for waste detection and image stitching.

A study adopted a novel approach that used convolutional neural networks (CNNs) for image stitching and compared the results with those of traditional methods.[3] Although this approach improved stitching quality, it has notable limitations, including the need for large datasets, long training times, and limited applicability to fixed scenarios. Moreover, it cannot operate in real time.

Drones have been employed to perform near-real-time image stitching.[4] Several feature detectors were evaluated: Binary Robust Invariant Scalable Keypoints (BRISK), Scale-Invariant Feature Transform (SIFT),[5] unmodified SIFT, Oriented Features from Accelerated Segment Test (FAST) and Rotated Binary Robust Independent Elementary Features (BRIEF) (ORB),[6] and Accelerated KAZE (A-KAZE).[7] The study compared each method's characteristics, computational efficiency, and qualitative performance. The study concluded that ORB and A-KAZE were the most suitable for real-time image stitching due to their lower computation times.

A study found that relying solely on traditional feature point extraction algorithms results in poor matching performance in image stitching.[8] To address this issue, four feature extraction methods were proposed—ORB, BRISK, SIFT, and Speeded-Up Robust Features—along with the brute-force (BF) matcher[9] and distance-based matching. Although this method can be time-consuming due to the large number of required comparisons, it yields higher matching accuracy. Another study proposed using SIFT for feature extraction together with the Fast Library for Approximate Nearest Neighbors (FLANN) for nearest-neighbor feature matching.[10,11] This approach enhances matching performance and processes large datasets more efficiently, making FLANN more suitable for real-time applications than the BF matcher.

An improved ORB algorithm for aerial image reconstruction was introduced.[12] The improved ORB algorithm was used for feature extraction and description, followed by coarse feature matching using the K-nearest neighbors (KNN)[13] algorithm. Incorrect matches were subsequently removed using Random Sample Consensus (RANSAC).[14] Compared with traditional ORB and SIFT, the improved ORB significantly improved stitching quality and produced more effective aerial image reconstruction.

An optimized SIFT algorithm was proposed to address the uneven distribution of feature points often encountered in image stitching.[15] The method segments the image to divide feature regions and ensures that the number of feature points in each region remains below a predetermined threshold before matching. Coarse matching is performed using KNN, and RANSAC is applied to eliminate mismatches, thereby improving stitching accuracy. When SIFT and Speeded-Up Robust Features are combined with RANSAC, they can achieve high-accuracy feature matching, but these traditional methods are computationally expensive and unsuitable for real-time processing. To overcome this limitation, a method that uses ORB together with Progressive Sample Consensus (PROSAC)[16] was proposed.[17] ORB reduces computational complexity and accelerates processing, while PROSAC improves upon RANSAC's randomness, consistently lowering computation time and enhancing overall performance, thereby enabling real-time image stitching.

A hybrid fusion algorithm based on pyramid decomposition was proposed to first decompose the image into a Gaussian pyramid and then compute differences to construct a Laplacian pyramid.[18] Different fusion strategies were applied at each decomposition level to achieve multi-layer fusion. This method effectively handles brightness variations in image details and preserves color richness, producing fusion results with no ghosting and smooth, natural transitions. However, the multi-scale spatial decomposition increases computational complexity, making the processing workflow more demanding.

To further improve the quality of image stitching, an enhanced gradual fusion algorithm was introduced.[19] It can produce seamless stitches and achieve smooth transitions at seam boundaries. To obtain accurate Global Navigation Satellite System (GNSS) coordinates and UAV heading angles, a study employed two identical GNSS receivers in Real-Time Kinematics (RTK)

mode. This configuration achieves centimeter-level positioning accuracy and provides precise heading estimates for the UAV.[20]

Another study used YOLOv8[21] for waste detection and classification to improve the efficiency and safety of waste management.[22] Their results demonstrated that YOLOv8 can accurately detect and classify various types of waste. Instance segmentation, which integrates object detection and semantic segmentation, was also discussed as a method for identifying pixel-level regions of interest and delineating object boundaries. YOLOv8 and Mask R-CNN were applied for instance segmentation in diverse orchard environments.[23] Their findings showed that YOLOv8 achieved outstanding accuracy and near-perfect recall, surpassing Mask R-CNN.

By integrating the above techniques, the system enhances the precision of UAV GNSS coordinates and stabilizes heading angle estimation. The YOLO-based real-time UAV-based riverine waste monitoring system is then used to detect waste. Combined with advanced methods for feature extraction, feature matching, mismatch elimination, and seam fusion, the system can perform real-time dynamic image stitching. In this study, we applied the real-time UAV riverine waste monitoring system[1] to YOLO-based object detection (YOLOv5 and YOLOv8) and instance segmentation to identify pollution, and compared their training performance and detection results. The system also supports real-time dynamic image stitching, which involves feature detection and description, feature matching, homography matrix estimation, and seam fusion.

Feature detection and description are optimized using SIFT, ORB, and AKAZE. Feature matching is performed with the BF matcher using KNN, and with FLANN using KNN matching. Homography estimation utilizes RANSAC and PROSAC to remove mismatches. Seam fusion uses a weighted masking technique to remove seams and color inconsistencies in stitched images. Together, these methods integrate 12 distinct image-stitching combinations, enabling a comprehensive comparison of stitching results. In this study, a real-time riverbank garbage identification system using multiple UAVs is proposed; optimal image stitching methods are acquired; and the size of the waste area, type of waste, and current coordinates of waste location are obtained.

2. System description

The real-time UAV-based riverine waste monitoring system is a versatile system and comprises nine components.

2.1. Methods

The sender and receiver were configured on laboratory computer servers, ensuring a reliable data connection. Images were transmitted to the Kafka Video Streaming Server, a platform for real-time data streaming. Kafka is a publish/subscribe messaging system. It is used as a distributed event streaming platform in many applications, such as financial transactions, automotive industry tracking, and business customer interactions. Kafka's communication protocol is transmission control protocol, the architecture is peer-to-peer, and the queuing model is Pub/Sub. Kafka efficiently handles multiple producers and multiple consumers, and can scale up the brokers to handle large volumes of data. Kafka's disk-based retention ensures that, even under load, there is no risk of data loss. The data are retained in Kafka with custom retention rules.

In the UAV garbage detection task, the UAV continuously consumes messages from the Kafka topic to check for any commands from the control station. Each UAV consumes one partition in the topic. The database supports large volumes of data for storage and querying. In this study, NoSQL databases are faster than SQL databases, and MongoDB demonstrates the best performance. We selected MongoDB as our database due to its high performance and schema-less data storage capability. MongoDB is a versatile data storage system and is used for data storage and real-time monitoring, with results displayed on a website. In MongoDB, a collection is created to store documents. The garbage information sent by the UAV includes the UAV ID, the time at which garbage is detected, the garbage's latitude and longitude, and the area affected by the waste.

Figure 1 shows the monitoring system, a versatile tool for monitoring riverine waste. The sender, equipped with an embedded system and a 4G network, enables versatile transmission of images, the UAV's GNSS coordinates, and garbage information to the server. This system also supports real-time image stitching, as shown in the UAV sender architecture in **Figure 2**. The

flight-controller platform is based on Pixhawk. The core stabilization and navigation loops are built around proportional–integral–derivative control. The proportional–integral–derivative controller computes a control output based on the error between the desired and measured states. Before applying control rules, Pixhawk firmware estimates the vehicle's state using sensor fusion. The estimator combines gyroscope, accelerometer, magnetometer, global positioning system (GPS), and barometer sensors to obtain accurate estimates of orientation, position, and velocity. In attitude control, it includes control of roll, pitch, and yaw angles. It compares the desired attitude with the inertial measurement unit-estimated attitude and then outputs desired angular rates. In rate control, angular velocities are controlled using gyroscope measurements, and motor commands (throttle adjustments) are then output. In position and velocity control (for autonomous flight), it controls GPS position, altitude, and velocity using GPS, barometer, and inertial measurement unit data and then generates desired attitudes and thrust commands. This layered structure improves stability and responsiveness.

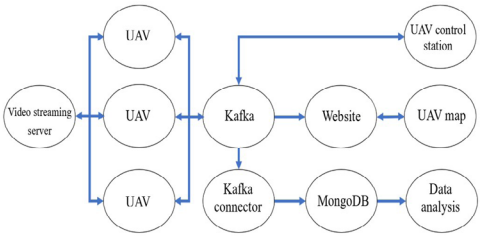

Figure 1. The architecture of the real-time unmanned aerial vehicle (UAV)-based riverine waste monitoring system

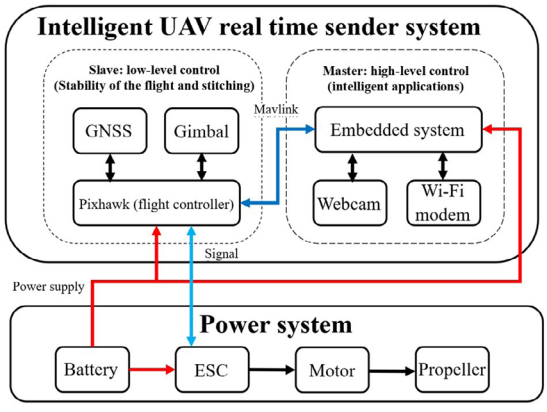

Figure 2. The architecture of the UAV sender
Abbreviations: ESC: Electronic speed controller; GNSS: Global Navigation Satellite System; UAV: Unmanned aerial vehicle.

2.2. Hardware

The video equipment uses a USB Video Class (UVC) webcam, such as the Logitech BRIO.[24] The webcam is mounted on the UAV's two-axis gimbal to ensure image stability. The Wi-Fi Modem is a crucial component that provides network connectivity for the embedded system. It also plays a key role in transmitting the UAV's GNSS coordinates, waste-identification results, pollution areas, and real-time images to the laboratory's computer server. Image transmission is handled via imageZMQ,[25] while the remaining data is streamed via Kafka. The Logitech webcam is mounted on the UAV's gimbal to maintain image stability. The Pixhawk series is used to control Hexacopters (**Figure 3** and **4**) and Octocopters (**Figure 5**). Peripheral equipment includes an SIYI receiver, a telemetry radio, GNSS, motors, electronic speed controllers, and batteries, with motor and electronic speed controller specifications and quantities selected based on specific requirements.

The GNSS receiver uses PMGN1 with a dual-frequency GNSS antenna, an electronic compass, and a low-cost, high-precision dual-frequency GNSS receiver chip, the U-blox F9P. This receiver employs RTK technology to obtain accurate GNSS coordinates. The embedded system uses the NVIDIA Jetson H2.0, which offers advantages such as a small form factor, a lightweight design, and high performance. It can be employed for pollution identification and for data and image reception and transmission on UAVs. The gimbal is a two-axis, brushless, lightweight, computer numerical control-machined aluminum gimbal designed to precisely control posture. It allows precise pitch-angle adjustments via pulse width modulation signals via radio control.[26]

Figure 3. The Hexacopter_1

Figure 4. The Hexacopter_2

Figure 5. The Octocopter

3. Image stitching

This section presents a feature-based image stitching algorithm. The process begins with feature detection and description of the input images, followed by feature point matching across three overlapping images. Then, the homography matrix is computed to perform reprojection and produce the stitched image. Finally, the weighted mask method is applied to remove seams and correct color discrepancies in the stitched result.

3.1. Feature detection and description

This subsection explores various feature detection and description methods employed individually in image stitching, including SIFT, ORB, and AKAZE.

3.1.1. Scale-invariant feature transform

Scale-invariant feature transform[5] has advantages including scale invariance, rotation invariance, local feature extraction, rich gradient descriptors, high matching accuracy, strong robustness to noise, efficient matching efficiency, and partial invariance to illumination changes.

(a) Scale-space extrema detection and finding potential keypoints

Scale-invariant feature transform constructs a scale space to detect feature points across different scales and achieve scale invariance. While the Laplacian of Gaussian can construct a scale space, it incurs a high computational cost. Therefore, SIFT uses the difference of Gaussian (DoG) approximation to detect extreme points in the scale-space domain.

Gaussian blur is applied to image $I(x,y)$ to obtain images $L_G(x,y,\sigma)$ of different scales, where σ represents the scale, and x and y are the coordinates of image $I(x,y)$, as described in **Equation 1**, and "*" denotes the convolution operation in x and y. The Gaussian kernel function $G(x,y,\sigma)$ is defined in **Equation 2**. The DoG image $D(x,y,\sigma)$, shown in **Equation 3**, is formed by calculating the difference between Gaussian-blurred images at adjacent scales, with k being the scale multiplication factor, as shown in **Figure 6**.

To detect extremum points, a 3 × 3 × 3 neighborhood search is performed for each pixel in the DoG image, searching for extremum points in both space and scale, as shown in **Figure 7**. A pixel is considered a potential key point if it is the maximum or minimum among all 26 neighboring pixels in its 3 × 3 × 3 neighborhood.

$$L_G(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \tag{1}$$

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

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

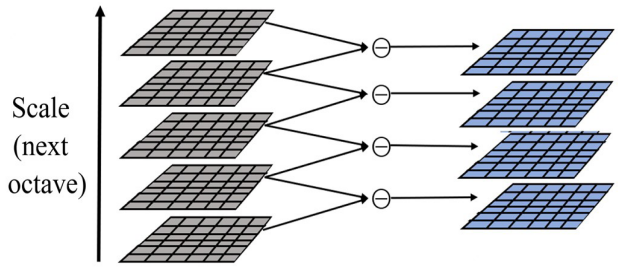

Figure 6. The architecture of the difference of Gaussian, modified from Lowe[5]

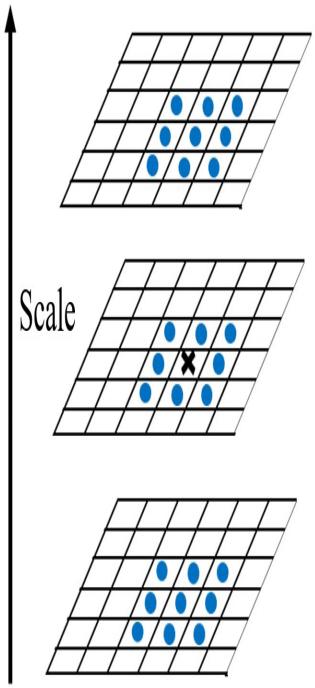

Figure 7. Maxima and minima result from the difference of Gaussian, modified from Lowe[5]

(b) Keypoint localization

Keypoint localization in SIFT involves precisely identifying potential keypoints and removing those with low contrast or strong edge responses. First, the DoG image $D(x, y, \sigma)$ undergoes a second-order Taylor series expansion around a keypoint to refine its position. This is given by **Equation 4**, where D is the value of the DoG at the sample point and $\mathrm{x}=(x,y,\sigma)^T$ represents the offset. The exact extremum position $\hat{\mathrm{x}}$, is obtained by solving for the derivative's zero point, resulting in **Equation 5**. Substituting $\hat{\mathrm{x}}$ back into the Taylor expansion yields the contrast threshold for the keypoint, and then **Equation 6** can be obtained. Keypoints with low contrast are removed if $|D(\hat{\mathrm{x}})|$ is below a certain threshold, which is set to 0.03.

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

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

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

To eliminate edge responses, the principal curvatures are computed from the 2 × 2 Hessian matrix H, as shown in **Equation 7**, where D_{xx} is curvature along the x-direction twice, D_{yy} is curvature along the y-direction twice, and D_{xy} is curvature along the x-direction and then the y-direction. The trace and determinant of H are calculated as in **Equations 8** and **9**, respectively, with $\alpha=\gamma\beta$, where γ is the ratio between the largest magnitude eigenvalue of H and the smaller one. If the ratio of the square of the trace to the determinant exceeds a specific value, given by **Equation 10**, the keypoint is discarded. Finally, filtering with **Equation 11** ensures that the selected key points are stable rather than merely edge responses.

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

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

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

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

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

(c) Orientation assignment

In the orientation assignment step of SIFT,[5] each keypoint is assigned one or more orientations to achieve rotation invariance. First, the gradient magnitude $M(x, y)$, as described in **Equation 12**, and orientation $\theta(x, y)$, as described in **Equation 13**, are calculated for each keypoint neighborhood. L_G is the Gaussian-smoothed image. Next, an orientation histogram with 36 bins, each covering 10 degrees, is constructed within the keypoint neighborhood. The histogram is weighted by gradient magnitudes and orientations, as shown in **Figure 8**. Its peak indicates the keypoint's primary orientation. If additional peaks exceed 80% of the primary peak, the keypoint is assigned multiple orientations to ensure robustness.

$$M(x,y)=\sqrt{M1+M2} \tag{12}$$

where

$$M1=\left(L_G(x+1,y)-L_G(x-1,y)\right)^2$$
$$M2=\left(L_G(x,y+1)-L_G(x,y-1)\right)^2$$

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

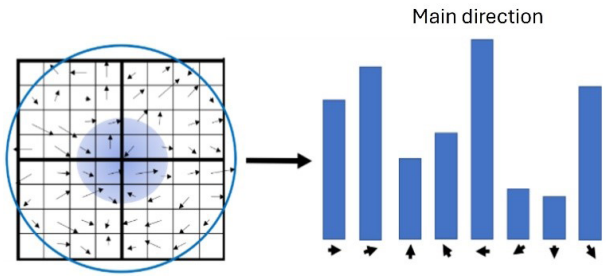

Figure 8. Histogram of main directions of keypoints

(d) Keypoint descriptor calculation

A local image patch around each keypoint is constructed to capture gradient information and generate feature vectors. First, gradients are computed in the rotated, scale-normalized coordinate system within the keypoint neighborhood. Then, a 16 × 16 neighborhood centered on the keypoint is selected and divided into 4 × 4 sub-regions. Each sub-region constructs an 8-bin orientation histogram, resulting in a 128-dimensional feature vector (4 × 4 × 8), as shown in **Figure 9**.

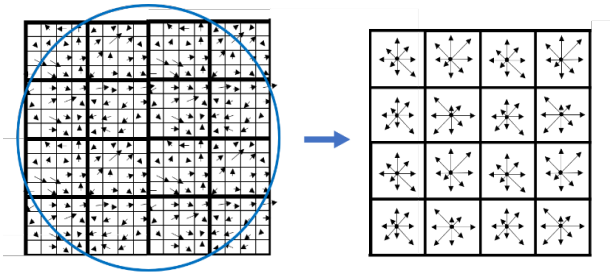

Figure 9. Image gradients and keypoint descriptor

3.1.2. Oriented FAST and Rotated BRIEF

The ORB is a fast algorithm for feature point extraction and description, divided into two parts: extraction and description. Feature extraction is developed from the FAST algorithm,[27] while feature description is improved based on the BRIEF feature descriptor algorithm.[28] ORB features combine the FAST feature point detector with the BRIEF descriptor and have been further improved and optimized based on their original foundations.

(a) Features from Accelerated Segment Test

The first step in ORB[6] is to use the FAST algorithm to detect keypoints. For each pixel, consider the 16 pixels on a circle with a radius of 3 centered at that pixel. Suppose there are 12 contiguous pixels among these 16 with intensity values either higher than the intensity of pixel p plus a threshold t or lower than the intensity of pixel p minus the threshold t, then the pixel is considered a feature point.

The algorithm's quick decision steps are as follows: for each candidate pixel p, select the 16 pixels on a circle of radius 3 centered at p, set an intensity threshold t, and examine the four critical pixels on the circle (usually the 1st, 5th, 9th, and 13th). If three or more of these critical pixels have values either greater than the intensity of p plus t or less than the intensity of p minus t, then proceed to further examination. Otherwise, discard the pixel p, as shown in **Figure 10**.

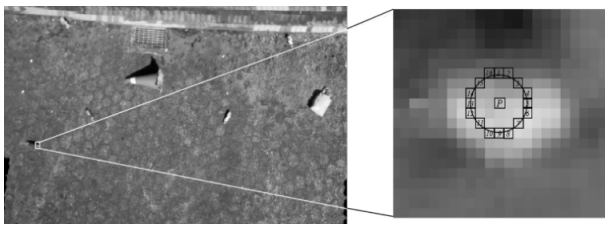

Figure 10. Example of image sampling

(b) Features from Accelerated Segment Test keypoint orientation

To ensure the descriptors are rotationally invariant, ORB computes the direction of each keypoint using the intensity centroid method. First, it calculates the moments m_{pq} of the region around the keypoint, as shown in **Equation 14**. The moment m_{pq} is obtained by summing the weighted intensities of all pixels (x, y) in the region, where $I(x, y)$ represents the intensity of the pixel (x, y), and p and q are the weighting exponents used to adjust each pixel's contribution to the moment. Specifically, m_{00} is the sum of the intensity values of all pixels in the region, m_{10} is the sum of the intensity values weighted by the horizontal position x, and m_{01} is the sum of the intensity values weighted by the vertical position y. Then, the centroid C is calculated based on these moments, as shown in **Equation 15**. Finally, the direction θ of the keypoint is defined as the direction of the vector from the keypoint to the centroid, as shown in **Equation 16**. These steps enable calculation of each keypoint's direction, thereby providing the descriptors with rotational invariance. To achieve greater rotational invariance, moment calculations are restricted to points (x,y) located within a circular region of a predetermined radius.

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

$$C = \begin{pmatrix} \frac{m_{10}}{m_{00}} & \frac{m_{01}}{m_{00}} \end{pmatrix} \tag{15}$$

$$\theta = atan2\left(m_{01}, m_{10}\right) \tag{16}$$

(c) Rotated Binary Robust Independent Elementary Features

Rotated BRIEF[4] performs Gaussian smoothing using a 9 × 9 Gaussian convolution kernel to filter and reduce noise, thereby somewhat alleviating noise sensitivity. The Rotated BRIEF computes the binary feature descriptor as a binary string within a local region (Patch) of size S × S around the feature point by randomly selecting several pairs of points (a, b) within this region and comparing their grayscale values. The grayscale test τ is defined as **Equation 17**, where $P(a)$ and $P(b)$ are the pixel intensities at locations $a = (u_1, v_1)$ and $b = (u_2, v_2)$ respectively, after the points are smoothed by Gaussian filtering. Randomly select n pairs of random points to form a binary code. This binary code describes the feature points, known as the feature descriptor, as shown in **Equation 18**, where $n = 256$ is commonly used.

$$\tau\left(P; a, b\right) = \begin{cases} 1 : P(a) < P(b) \\ 0 : P(xa) \geq P(b) \end{cases} \tag{17}$$

$$f_n\left(P\right) = \sum_{1 \leq i \leq N} 2^{i-1} \tau\left(P; a_i, b_i\right) \tag{18}$$

To make BRIEF robust to in-plane rotation, the keypoint orientation is used to steer it. For any set of n binary tests at locations (x_i, y_i), a 2 × n matrix is defined as shown in **Equation 19**. Using the angle θ and the rotation matrix R_θ obtained by FAST, as shown in **Equation 20**, the rotated coordinate matrix is given by **Equation 21**. This process ensures that the BRIEF descriptors are rotation-invariant by aligning them with the orientation of keypoints detected by FAST.

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

$$R_\theta = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \tag{20}$$

$$S_\theta = R_\theta S \tag{21}$$

3.1.3. Accelerated-KAZE

Accelerated KAZE[7] is an accelerated version of KAZE that uses fast explicit diffusion (FED) to speed up feature detection in nonlinear scale space.

(a) Nonlinear diffusion filtering

KAZE exploits the nonlinearity of the underlying differential equations to diffuse an image's brightness into a nonlinear scale space. In the classical nonlinear diffusion **Equation 22**, *div* and ∇ are the divergence and gradient operators, respectively, t is the scale parameter, with larger values yielding simpler image representations, and L denotes the image's brightness. **Equation 23** introduces the conductivity function c into the diffusion equation, enabling the diffusion process to adapt to the local image structure, where the function ∇L_G is the gradient of a Gaussian-smoothed version of the original image L. The conductivity function g is given in **Equation 24**, and the parameter λ is a contrast factor that controls the diffusion level, determining which edges need to be enhanced or preserved—a higher value results in fewer preserved edge details.

$$\frac{\partial L}{\partial t} = div\left(c\left(x, y, t\right) \cdot \nabla L\right) \tag{22}$$

$$c\left(x, y, t\right) = g\left(\left|\nabla L_G\left(x, y, t\right)\right|\right) \tag{23}$$

$$g = \frac{1}{1 + \frac{\left|\nabla L_G\right|^2}{\lambda^2}} \tag{24}$$

(b) Fast explicit diversion

The motivation behind FED is to use explicit schemes to decompose box filters. Iterated box filters can approximate Gaussian kernels well and are easy to implement. The main idea is to perform several cycles, each consisting of n explicit diffusion steps with step sizes τ_j derived from the decomposition of the box filter as in **Equation 25**, where τ_{max} is the maximum step size that does not violate the stability condition of the explicit scheme. The stopping time, θ_n, of one FED cycle is shown in **Equation 26**. The FED cycle has n variable step sizes, τ_j, and is given in **Equation 27**, where $A(L^i)$ is a matrix that encodes the conductivities for the image.

$$\tau_j = \frac{\tau_{max}}{2cos^2\left(\pi \frac{2j+1}{4n+2}\right)} \tag{25}$$

$$\theta_n = \sum_{j=0}^{n-1} \tau_j = \tau_{max} \frac{n^2 + n}{3} \tag{26}$$

$$L^{i+1,j+1} = \left(I + \tau_j A\left(L^i\right)\right) L^{i+1,j} \quad j = 0,\ldots,n-1 \tag{27}$$

(c) Building a nonlinear scale space with fast explicit diversion

When building a nonlinear scale space, a set of evolution times is defined to construct it. The scale space is discretized into a series of O octaves and S sub-levels, with the corresponding scale σ (in pixels) mapped through **Equation 28**, where N is the total number of filtered images. Since nonlinear diffusion filtering operates in time units, the discrete scale levels σ_i in pixel units are converted to time units, as shown in **Equation 29**.

$$\sigma_i(o,s) = 2^{o+\frac{s}{S}}, \tag{28}$$

$$o \in [0\ldots O-1], s \in [0\ldots S-1], i \in [0\ldots N]$$
$$t_i = \frac{1}{2}\sigma_i^2, i \in [0\ldots N] \tag{29}$$

(d) Feature detection in Accelerated KAZE

For feature detection in AKAZE, for each filtered image L^i in the nonlinear scale space, the determinant of its Hessian matrix is calculated as in **Equation 30**, where $\sigma_{i,norm}^2 = \left(\frac{\sigma_i}{2^{o^i}}\right)^2$ is the scale after scale normalization, L^i is the filtered result of the image at the i-th scale, and L_{xx}, L_{yy}, L_{xy}, and L_{yx} are the elements of the Hessian matrix. The local maxima of the normalized Hessian matrix at different scales are used to extract feature points.

$$L_{Hessian}^i = \left(\frac{\sigma_i}{2^{o^i}}\right)^2 \left(L_{xx}^i L_{yy}^i - L_{xy}^i L_{yx}^i\right) \tag{30}$$

3.2. Feature point matching

This subsection introduces three feature point matching methods: the BF matcher, the FLANN-based matcher, and the KNN matching method.

3.2.1. Brute-force matcher

The BF matcher[9] is a fundamental feature-matching method suitable for small-scale or simple feature-matching problems. Its advantages lie in its simplicity and accuracy, while its disadvantages include high computational load and slow speed. The basic process involves comparing each descriptor in the first image to all descriptors in the second image, computing the distance between each pair, and selecting the pair with the smallest distance as the best match. This process is repeated continuously until all descriptors have been matched.

3.2.2. Fast Library for Approximate Nearest Neighbors-based matcher

The FLANN[14] is a library designed specifically for fast nearest-neighbor searches and high-dimensional feature matching. It focuses on approximate nearest neighbor search, providing multiple indexing structures and automatic parameter tuning, making it ideal for handling large-scale, high-dimensional data. The efficiency of nearest-neighbor searches can be significantly improved while maintaining a certain level of accuracy. Allowing for some error to speed up searches is crucial when dealing with large-scale data in high-dimensional spaces, as exact nearest-neighbor searches are computationally intensive in such settings. It works faster than the BF matcher for large datasets.

3.2.3. K-nearest neighbors matching

The KNN algorithm[13] extracts keypoints and computes descriptors from two images, yielding two descriptor sets. For each descriptor in the first set, it calculates distances to all descriptors in the second set and finds the K closest. To improve the accuracy of the matches, a threshold is used for filtering; the nearest and second-nearest distances are compared, and only if the ratio of the smallest distance to the second smallest distance is less than the preset threshold is the match considered reliable (typically, K is set to 2, and the threshold is set to 0.7).

3.3. Homography matrix and eliminate mismatch points

The homography matrix is primarily used to achieve coordinate transformation between images in image stitching. First, feature points in the images are detected and matched to find corresponding point pairs. Then, using these corresponding points, the homography matrix is estimated to transform the coordinate system of one image to that of the other. Finally, this matrix is applied to perform a perspective transformation on the image, and the transformed image is stitched with the target image. In **Equation 31**, the matrix H_m transforms the points (x_i, y_i) from the right image to the points (x_i', y_i') in the left image.[29] The homography matrix is a 3 × 3 matrix, but with 8 degrees of freedom, as it is estimated up to a scale.

$$s_i \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = H_m \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad (31)$$

Many mismatches often occur when detecting and matching feature points in images. To improve the accuracy of homography matrix estimation, methods to eliminate these mismatches are necessary. Therefore, this section introduces two methods for removing mismatched points: random sample consensus (RANSAC) and progressive sample consensus (PROSAC).

3.3.1. Random sample consensus

Random sample consensus[15] is highly robust; even when the data contains many outliers, it can still accurately estimate model parameters. It is relatively simple, easy to implement, and suitable for a wide range of model-fitting problems. However, it also has some disadvantages: it requires multiple iterations, leading to high computational costs when processing large datasets, and it employs random sampling, which may yield a suboptimal solution rather than the global optimal one. The steps are given as follows:

(i) Step 1: Randomly select four pairs of matching points and compute the homography matrix *H*.
(ii) Step 2: Map all points from the first to the second image and calculate the reprojection error; if the error is less than the threshold, then add the inlier point, *X*.
(iii) Step 3: If the current number of inliers, *X*, is greater than the number of inliers in the best model, *Y*, then update the best model *Y* = *X*.
(iv) Step 4: Repeat steps 1 to 3 until the maximum number of iterations, *n*, is reached or the number of inliers meets the condition.

Finally, use the best set of inliers to re-estimate the final homography matrix *H*.

3.3.2. Progressive sample consensus

Progressive sample consensus[17] is an improved version of RANSAC. By introducing a progressive sampling strategy, it prioritizes sampling from higher-quality data points and gradually extends to include more data points, repeatedly verifying the model. This approach allows PROSAC to converge to the optimal model more quickly, enhancing the efficiency and robustness of model estimation, particularly in datasets with significant noise and outliers.

3.4. Image fusion

In the pyramid decomposition hybrid fusion algorithm,[18] the main idea is to decompose an image into multiple frequency components, merge at each frequency level, and finally reassemble it into the final image. The main steps of the pyramid-decomposition image-stitching fusion algorithm are as follows:

(i) Step 1 (construct the Gaussian pyramid): This involves downsampling the image in stages using a Gaussian filter to smooth it and gradually remove high-frequency components. Each level of the pyramid image becomes smoother while retaining low-frequency information. The Gaussian filter is given by **Equation 32**, where $G(x,y,\sigma)$ denotes the Gaussian function, and σ is the standard deviation. Downsampling is described in **Equation 33**, where I_k represents the image at level k, and I_{k+1} represents the image at level $k + 1$.
(ii) Step 2 (construct the Laplacian pyramid): The Laplacian pyramid represents the high-frequency components of the image. Each level of the Laplacian pyramid image is the difference between the Gaussian pyramid image and the upsampled version of the next level of the Gaussian pyramid. This way, each level of the Laplacian pyramid mainly retains the high-frequency information of that layer. Gaussian upsampling is described in **Equation 34**, where I_{up} is the upsampled image. Laplacian layer calculation is described in **Equation 35**, where L_k represents the Laplacian layer at level k, I_k is the Gaussian image at level k, and I_{up} is the upsampled image from I_k.
(iii) Step 3: Fuse the corresponding pyramid levels of each image. The fusion method can be a weighted average, a maximum, a minimum, or another method, depending on application requirements. A standard procedure is weighted fusion, where a weight map determines each image's contribution.
(iv) Step 4 (reconstruct the image): This step involves reassembling the fused Laplacian pyramid levels into the final image, starting with the lowest-frequency level and progressively moving to higher ones, ultimately yielding the complete fused image.

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (32)$$

$$I_{k+1}(x,y) = I_k(2x,2y) \quad (33)$$

$$I_{up}(x,y) = I_k\left(\frac{x}{2},\frac{y}{2}\right) \tag{34}$$

$$L_k = I_k - I_{up} \tag{35}$$

However, the decomposition process is relatively complex, as the algorithm constructs the image's multi-scale space using Gaussian and Laplacian pyramids. Additionally, using different fusion algorithms to merge the high-frequency and low-frequency components of the decomposed image further increases the computational complexity. As shown in **Figure 11**, the stitching and fusion process is prone to errors and failures. Therefore, the weighted fusion method is adopted, which is more straightforward and feasible for real-time applications.

Figure 11. The stitching and fusion failures

Weighting mask methods for fusing images at the seams[19] primarily use a linear transformation based on the weight, defined as the ratio of a pixel's position within the overlapping area to the width of the overlapping area's boundary. X_2 is the length of image A, $X_3 - X_1$ is the length of image B, and $X_2 - X_1$ is the overlapping area of the images. If the overlapping area X is closer to X_2, the weight W_A of image A in the fusion is smaller, as shown in **Equation 36**, and the weight W_B of image B in the fusion is more significant, as shown in **Equation 37**. The grayscale values of the overlapping area can be computed and used for image fusion to eliminate splicing seams, as shown in **Equation 38** and **Figure 12**.

$$W_A = \frac{(X_2 - X_1) - (X - X_1)}{X_2 - X_1} = \frac{X_2 - X}{X_2 - X_1} \tag{36}$$

$$W_B = 1 - W_A \tag{37}$$

$$Z(x,y) = W_A * A(x,y) + W_B * B(x,y) \tag{38}$$

The averaging-weighted mask can also be derived as the solution of a least-squares optimization problem. Consider **Equation 38**; the optimal weight W_A can be obtained by minimizing the following loss function

$$J(W) = \sum_{(x,y)} [Z - X]^2 \tag{39}$$

where X is the desired reference. Substituting Z into **Equation 39**,

$$J(W) = \sum_{(x,y)} [W_A A + W_B B - X]^2 \tag{40}$$

Since $W_B = 1 - W_A$, thus

$$J(W) = \sum_{(x,y)} [W_A A + (1 - W_A) B - X]^2 \tag{41}$$

Taking the derivative with respect to W_A,

$$\frac{\partial J}{\partial W_A} = 2\sum_{(x,y)} [W_A A + (1 - W_A) B - X] A \tag{42}$$

Setting

$$\frac{\partial J}{\partial W_A} = 0 \tag{43}$$

Gives

$$W_A^* = \frac{\sum_{(x,y)} (X - B) A}{\sum_{(x,y)} (A - B) A} = \frac{\sum_{(x,y)} (B - X)}{\sum_{(x,y)} (B - A)} \tag{44}$$

This provides the least-squares optimal blending coefficient. At a given position where A is X_1, B is X_2, and the desired reference is X, then W_A can be obtained as in **Equation 36**, which can then be optimized to improve image stitching quality.

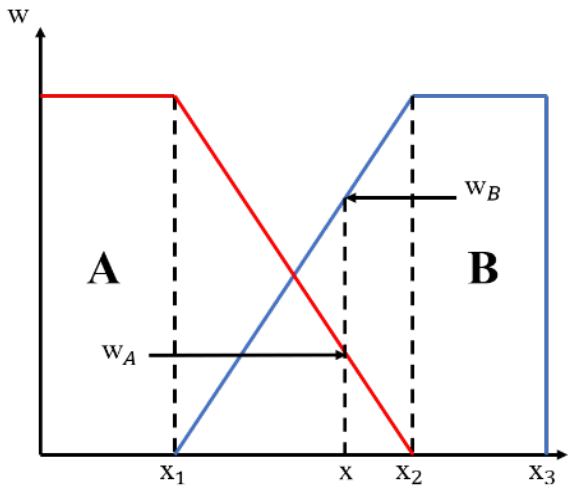

Figure 12. Weighted mask fusion images

3.5. Real-time image stitching test

The image stitching process is shown in **Figure**

13. The test was conducted indoors using a webcam to perform real-time image stitching in the same environment, as shown in **Figure 14**. Feature detection and description techniques used include SIFT, ORB, and AKAZE. For feature point matching, a BF matcher with KNN matching and an FLANN matcher with KNN matching were employed. RANSAC and PROSAC were utilized to find the homography matrix and eliminate mismatches. Image fusion was performed using a weighted mask fusion method.[19] Combining these methods yielded 12 distinct combinations. **Figure 15** shows the image-stitching results obtained using combinations of SIFT, BF, and RANSAC. **Figure 16** shows the result using the ORB, BF method, and RANSAC. **Table 1** presents the number of matched feature points and the total time spent by different combinational methods. All combinations can successfully stitch images, but the ORB–BF–RANSAC combination has fewer matched feature points and a shorter computation time. SIFT has highly accurate matched points, ORB has few matched points, and AKAZE has moderate matched points. In terms of speed, ORB is fast, SIFT is slow, and AKAZE is medium. For real-time UAV applications, ORB is the best choice. These images are stitched directly, without translation or rotation, during the image stitching process.

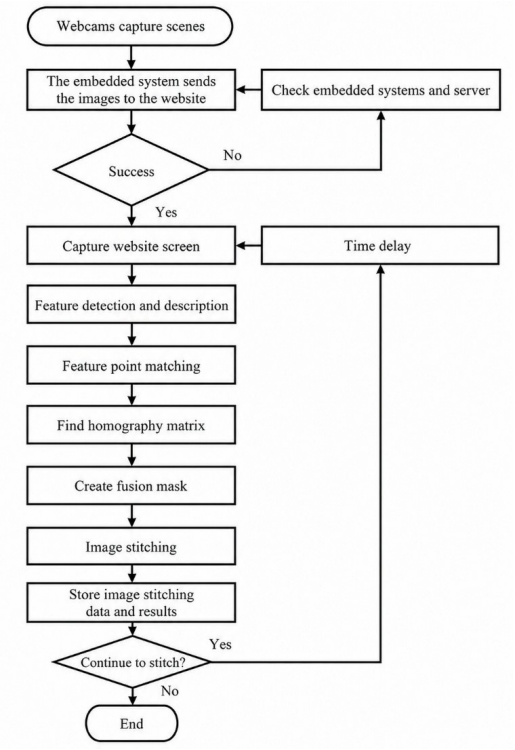

Figure 13. Image stitching process

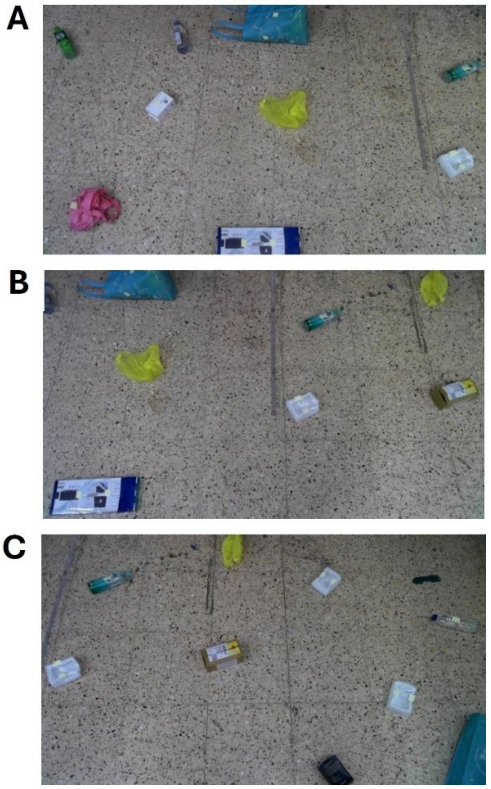

Figure 14. Original three images before stitching: (A) Left view, (B) middle view, and (C) right view

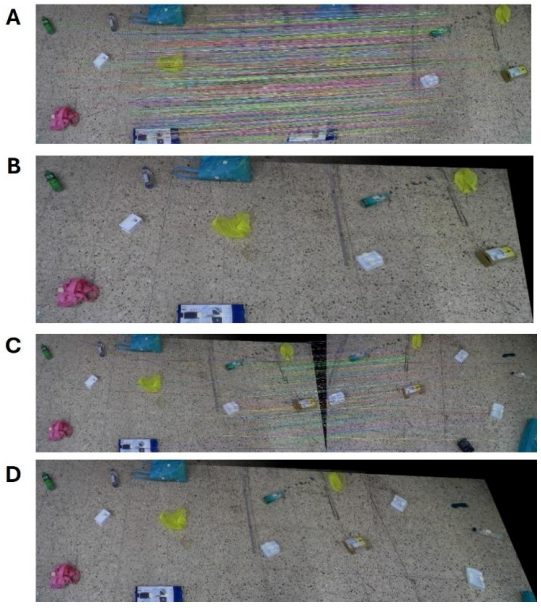

Figure 15. The SIFT–BF–RANSAC fusion image stitching process. (A) Feature points of the left and middle images. (B) Left and middle image stitching results. (C) Matching result of (B) and the right image feature points. (D) Image stitching results.
Abbreviations: BF: Brute-force; RANSAC: Random sample consensus; SIFT: Scale-Invariant Feature Transform.

Figure 16. The ORB–BF–RANSAC fusion image stitching
Abbreviations: BF: Brute-force; ORB: Oriented Features from Accelerated Segment Test and Rotated Binary Robust Independent Elementary Features; RANSAC: Random sample consensus.

Table 1. Image stitching feature point data

Method	First matched feature points	Second matched feature points	Total time spent (s)
SIFT–BF–RANSAC	1,499	180	7.75
SIFT–BF–PROSAC	1,250	46	7.71
SIFT–FLANN–RANSAC	1,264	74	7.00
SIFT–FLANN–PROSAC	1,282	61	6.48
ORB–BF–RANSAC	852	74	2.10
ORB–BF–PROSAC	852	72	2.92
ORB–FLANN–RANSAC	886	107	2.35
ORB–FLANN–PROSAC	876	90	2.23
AKAZE–BF–RANSAC	1,062	37	6.09
AKAZE–BF–PROSAC	1,062	31	7.25
AKAZE–FLANN–RANSAC	1,060	39	5.14
AKAZE–FLANN–PROSAC	1,065	35	5.56

Abbreviations: AKAZE: Accelerated KAZE; BF: Brute-force; FLANN: Fast Library for Approximate Nearest Neighbors; ORB: Oriented Features from Accelerated Segment Test and Rotated Binary Robust Independent Elementary Features; PROSAC: Progressive sample consensus; RANSAC: Random sample consensus.

4. Riverbank pollution identification

In this study, various YOLO models were applied to image identification. In a previous study, YOLOv5 was successfully used to identify garbage.[30] Here, we utilized a more efficient model (YOLOv8) for the waste identification problem and compared its performance with that of YOLOv5.

4.1. YOLOv5

YOLOv5 is divided into five models of varying size and complexity—YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x—as illustrated in **Figure 17**. When the training datasets are input, adaptive image scaling is used to maintain consistent image dimensions. Additionally, data augmentation and adaptive anchor box calculations are applied to produce a more diverse set of training samples.

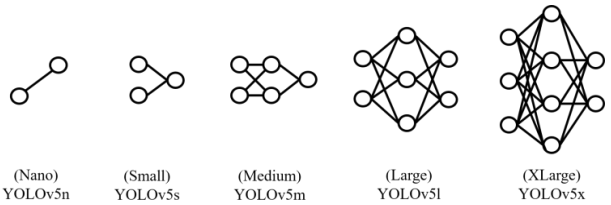

Figure 17. Diagrammatic representation of each YOLOv5 model (modified from Ref.[31])

4.1.1. Data augmentation techniques using YOLOv5

In YOLOv5 and YOLOv8, different data augmentation techniques are used to augment the training dataset. Methods include mosaic augmentation, copy-paste augmentation, random affine transformation, Mixup augmentation, albumentations, hue–saturation–value/brightness (HSV) augmentation, and random horizontal flip. A commonly used method is mosaic augmentation, which combines four images into a single image by placing each image in a quadrant of the new image. This technique helps the model learn to recognize objects in varying contexts and from different parts of an image.

Random affine transformation applies geometric transformations such as rotation, translation, scaling, and shearing to images, thereby making the model invariant to these transformations, thereby improving its robustness. The transformation parameters are randomly sampled within specified ranges to generate a variety of augmented images. Mixup augmentation creates new training examples by combining two images and their labels using a weighted sum, thereby smoothing the model's decision boundary by

encouraging it to behave linearly across training examples, thereby improving generalization and reducing overfitting.

Albumentations is a fast and flexible library for image augmentation. It provides a wide range of augmentation techniques, including but not limited to pixel-level transformations (e.g., brightness, contrast, and blur adjustments), spatial-level transformations (e.g., flipping, rotating, and cropping), and advanced techniques such as grid distortion and elastic transformation. HSV augmentation involves modifying an image's hue, saturation, and value (brightness) channels; this type of augmentation helps the model learn to recognize objects under varying lighting conditions and color intensities.

Random horizontal flipping flips the image with a certain probability; this technique helps augment the dataset by creating mirror images, making the model invariant to object orientation. Copy-paste augmentation is described in YOLOv5 as an innovative data enhancement method that involves copying objects from one image and pasting them into another, thereby generating new combinations of objects and backgrounds and improving the model's ability to generalize across different scenarios.[31]

Figure 18 shows the object detection training dataset after data augmentation, while **Figure 19** shows the instance segmentation training dataset after the same process. The bounding boxes and labels 0–2 in both figures correspond to the training categories: bottle, garbage, and the negative sample (rock).

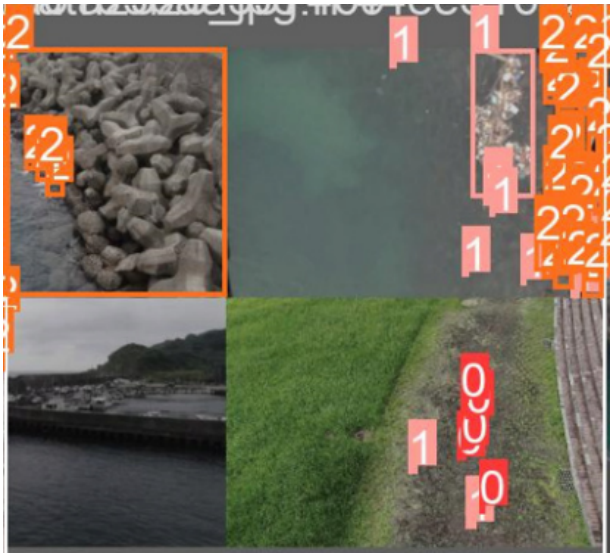

Figure 18. Object detection training dataset after data augmentation

Figure 19. Instance segmentation training dataset after data augmentation

4.1.2. Adaptive image scaling and anchor box calculation

In the training dataset, images typically vary in size, which poses a challenge for consistent model training. Adaptive image scaling mitigates this issue by resizing all images to a standard dimension. The process begins by calculating a scaling ratio from the original and target image sizes. Then, the images are resized, and any remaining areas are padded with a gray border. In YOLOv5, these gray borders are minimized to reduce unnecessary information, thereby improving training and inference efficiency.[32]

Adaptive anchor box calculation enhances detection performance by enabling the network to generate predicted boxes from initial anchor boxes, compare them with ground-truth boxes, compute differences, and iteratively adjust network parameters via backpropagation. This process involves extracting the widths and heights of target boxes from the annotation files, applying K-Means clustering to determine anchor boxes tailored to the dataset, and producing these anchor boxes, along with their average IoU, to further optimize the model's detection accuracy.[32]

4.1.3. Backbone, neck, and head layer in YOLOv5

In the backbone layer of YOLOv5_v7.0, the stem layer uses a ConvModule with a 6 × 6 kernel, replacing the Focus structure from previous versions. Except for the last stage layer, each layer consists of a ConvModule and a CSPLayer. The ConvModule includes a 3 × 3 Conv2d layer,

BatchNorm, and a SiLU activation function. The CSPLayer is a C3 module, which contains three ConvModules and several DarknetBottlenecks with residual connections. The last stage layer adds an SPPF module at the end. The SPPF module processes inputs sequentially through multiple 5 × 5 MaxPool2d layers, achieving similar effects to the SPP module but at a faster speed. Using the P5 model as an example, after Stages 2–4, feature maps are fed into the Neck structure. For a 640 × 640 input image, the output feature sizes are (256, 80, 80), (512, 40, 40), and (1,024, 20, 20), with strides of 8, 16, and 32, respectively.

In the neck layer, there are feature pyramid network (FPN) and path aggregation network (PAN) structures, both of which aim to address the limitations of multi-scale object detection by combining high-level semantic information with low-level spatial information to enable further extraction. In YOLOv5, FPN is implemented via upsampling and concatenation, whereas PAN is implemented via downsampling and concatenation.

The head layer comprises only three convolutional layers with non-shared weights to transform the input feature maps. The preceding PAN–FPN still outputs three feature maps of different scales, with shapes of (256, 80, 80), (512, 40, 40), and (1,024, 20, 20). Since YOLOv5 uses a non-decoupled output, tasks such as classification and bounding box detection are performed within different channels of the same convolutional layer.

4.2. YOLOv8

Figure 20 compares YOLOv8 with other real-time object detectors.[32] Although YOLOv8 is slower, it achieves better mean average precision (mAP).

This study primarily utilizes YOLOv8 for detection and segmentation. YOLOv8 has five main models: *n, s, m, l,* and *x*. Further details will be provided below.

4.2.1. Data augmentation techniques using YOLOv8

Regarding data augmentation, YOLOv8 is not significantly different from YOLOv5, as both use techniques such as mosaic, random affine transformation, Mixup, albumentations, HSV, and horizontal flip. YOLOv8 introduces the practice of disabling Mosaic during the last 10 epochs of training, as continuing this augmentation throughout the entire training process can reduce performance. The assumption that the training consists of 500 epochs is shown in **Figure 21**.[33]

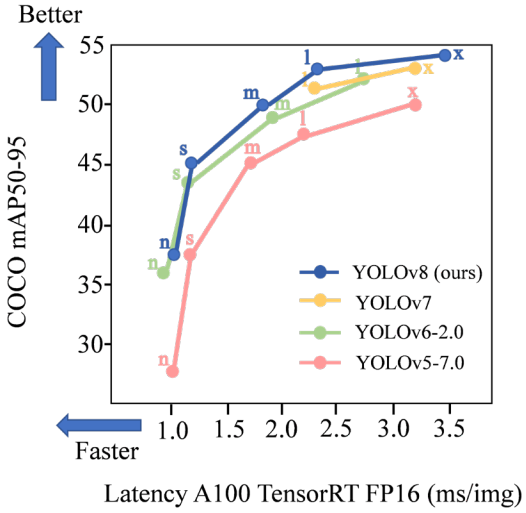

Figure 20. Comparison of various object detectors

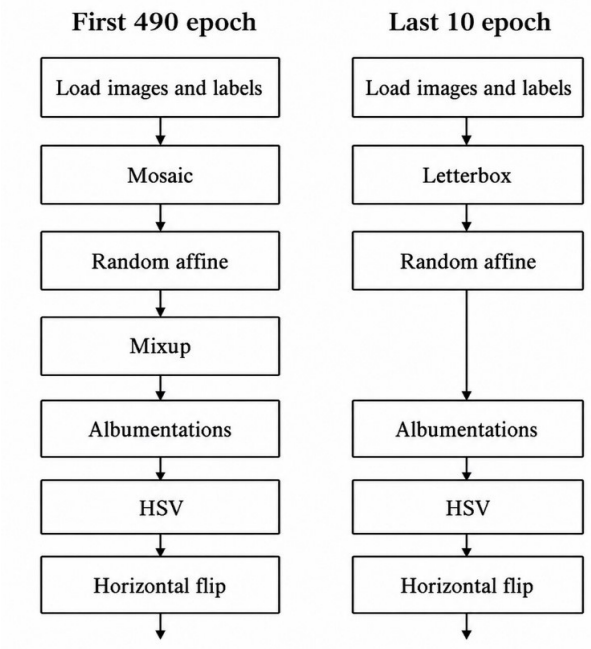

Figure 21. YOLOv8 train data augmentation architecture diagram
Abbreviation: HSV: Hue–saturation–value/brightness.

4.2.2. Backbone, neck, and head layer in YOLOv8

YOLOv8's backbone design, inspired by YOLOv7, replaces YOLOv5's C3 modules with C2f modules and adjusts channel counts across

different model scales, significantly enhancing performance. The backbone employs convolutional and deconvolutional layers for feature extraction, along with residual connections and bottleneck structures to reduce network size. The C2f unit offers fewer parameters and superior feature extraction compared to YOLOv5's C3. The C3 modules, previously configured in a 3-6-9-3 pattern, have been replaced by C2f modules in a 3-6-6-3 arrangement, introducing more skip connections and additional Split operations, as shown in **Figure 22**. Additionally, the kernel size of the first convolutional layer has been reduced from 6 × 6 in YOLOv5 to 3 × 3 in YOLOv8.[34]

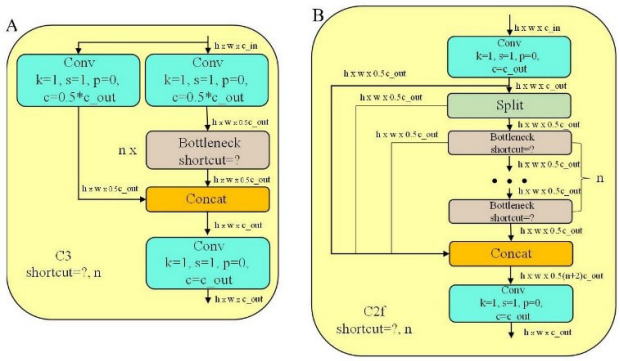

Figure 22. Comparison of (A) YOLOv5 and (B) YOLOv8 modules

YOLOv8 has removed two convolutional connection layers from the Neck module and adopted the PAN–FAN structure. The neck processes features from the backbone to enhance the accuracy and robustness of object detection. By incorporating various techniques, it merges multi-scale feature maps to better capture information across different scales. The YOLOv5 coupled head has been replaced by a YOLOv8 decoupled head, which no longer includes the previous objectness branch. The new decoupled head consists of a detection head and a classification head, responsible for the final tasks of object detection and classification.

4.2.3. Loss calculation

The loss calculation process consists of two parts: a positive and negative sample assignment strategy and loss computation. The loss computation includes the classification and regression branches, which omit the previous objectness branch. The classification branch uses binary cross-entropy loss, while the regression branch employs distribution focal loss and complete intersection over union (IoU) loss. The final loss is a weighted sum of these three individual losses.[35,36]

Considering the superiority of the dynamic assignment strategy, the YOLOv8 algorithm directly adopts the TaskAlignedAssigner. The pairing strategy of TaskAlignedAssigner can be summarized as selecting positive samples based on the weighted classification scores and box scores. Here, s is the predicted score corresponding to the annotated category, and u is the IoU between the predicted box and the ground-truth box. The product of these two values measures the alignment. An anchor box with an IoU greater than the threshold relative to the ground-truth box is considered a positive sample; otherwise, it is regarded as a negative sample.[37]

Anchor-based models use fixed anchor boxes to estimate the position and size of objects in images. These methods are typically slower because they require generating numerous key points and proposal boxes and performing complex classification and bounding box operations. To reduce model complexity, YOLOv8 adopts an anchor-free approach that does not rely on predefined anchor boxes but directly predicts object positions and categories, resulting in more accurate object detection.[33]

4.2.4. Segmentation

Instance segmentation is the computer vision task of recognizing objects in images and their associated shape. It is an extension of object detection, in which each prediction also includes a shape, rather than just a bounding box defined by a center point, width, and height. Instance segmentation can determine the number of objects in an image, the sizes of detected objects, their classifications, and their outlines.[38] The following section compares instance segmentation to other computer vision algorithms.

Semantic segmentation classifies each pixel in an image into a predefined category but does not differentiate between individual instances of the same class. Instance segmentation, on the other hand, identifies and separates each instance within a class, providing a more detailed and comprehensive understanding of the scene.

Instance segmentation and object detection models both identify the position of an object in an image, but object detection only predicts the object's bounding box, not its outline. Instance segmentation labeling is more laborious, and training accurate instance segmentation models is harder; these models are typically larger, slower,

and less optimized for edge deployment, and even instance segmentation models may require larger datasets to achieve the same accuracy as object detection models.

The network structures for segmentation and object detection have identical backbone networks and neck modules, differing only in the head. In the head module of the object detection model, additional segmentation branches have been added, including the mask coefficient branch and the prototype branch, as shown in **Figure 23**. The prototype branch outputs a set of mask prototypes in the first output layer, with each mask representing different mask information learned by the network; a single mask does not mean that there is only one target mask in the mask. The network outputs three pieces of information: class, bounding box, and mask coefficient. By parsing the first two, the target's class and bounding box can be predicted, allowing the corresponding mask coefficient to be obtained.[38]

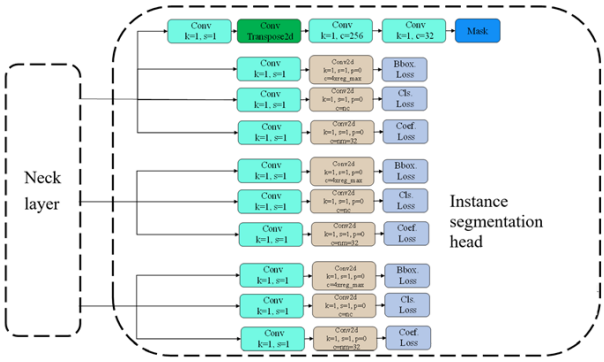

Figure 23. Instance segmentation network architecture (head)
Abbreviations: Bbox: Bounding box; Cls: Class.

The mask can be derived through a linear combination of the mask coefficients and the mask prototypes, where n represents the nth detection result, as shown in **Equation 45**.

$$mask_n = \sum_{i=0}^{32} mask\_coef_i * Proto_i \tag{45}$$

The mask within the detected bounding box is retained, and each pixel is thresholded to obtain the final mask for the target.

4.3. Pollution identification test

This section presents the training results of YOLOv8 using the PyTorch framework for object detection and instance segmentation using five model architectures (n, s, m, l, and x), three detection classes (bottle, garbage, and negative sample rock), and different training dataset sizes. However, the parameters could not be consistently adjusted for a detailed comparison due to computer limitations. The smaller the loss value, the better, and the primary purpose is to confirm whether the training has converged. The training results show that each loss has converged smoothly and successfully. After training for 1,000 epochs, mAP50(Box) and mAP50(Mask) show a downward trend.

We increased the training data to verify whether improved training results are achieved and to confirm whether overfitting has occurred. When the training datasets are increased to 3,657, mAP shows a smooth convergence trend. However, there is still some slight decrease and oscillation. The mAP50 is the mAP (average accuracy) at an IoU threshold of 0.5. This metric computes the average precision (AP) for all images in each category and then averages the APs across all categories, yielding mAP50. Precision is the ratio of correctly predicted positive observations to the total predicted positives. It indicates the level of false alarms in the identification process; a higher precision value means fewer false alarms. Recall is the ratio of correctly predicted positive observations to all the actual positives. It reflects the model's ability to identify true positives. A higher recall value indicates fewer false negatives.

In **Tables 2** and **3**, the overall performance can be analyzed using mAP50 (Box) and mAP50 (Mask). Based on the training results, the highest mAP50 (Box) is achieved by the YOLOv8m model with a training dataset of 3,657 images, while the YOLOv8n-seg model achieves the lowest. The YOLOv8x-seg model achieves the highest mAP50 (Mask), while the YOLOv8s-seg model achieves the lowest.

5. Experiments and results

This research utilized NVIDIA's embedded systems and a UAV-based riverine waste monitoring system to achieve real-time pollution identification and scene image stitching. The subsequent sections present and analyze the implementation results in detail.

5.1. Real-time image stitching process

Figure 24 presents the process of real-time image stitching in this study. The program combines three feature detection and description techniques, multiple feature point matching

Table 2. Comparison of training detection results

Model	Training datasets	Batch size	Precision	Recall	mAP50 (Box)
YOLOv8n	3,657	16	0.721	0.61	0.704
YOLOv8s	3,657	16	0.759	0.649	0.736
YOLOv8m	3,657	16	0.752	0.67	0.745
YOLOv8l	3,657	16	0.746	0.661	0.729
YOLOv8x	3657	16	0.745	0.626	0.705

Abbreviation: mAP: Mean average precision.

Table 3. Comparison of training segmentation results

Model	Precision (Box)	Precision (Mask)	Recall (Box)	Recall (Mask)	mAP50 (Box)	mAP50 (Mask)
YOLOv8n-seg	0.708	0.675	0.609	0.552	0.694	0.633
YOLOv8s-seg	0.744	0.699	0.618	0.546	0.712	0.629
YOLOv8m-seg	0.741	0.681	0.674	0.594	0.743	0.658
YOLOv8l-seg	0.742	0.698	0.656	0.57	0.727	0.648
YOLOv8x-seg	0.742	0.709	0.688	0.605	0.743	0.674

Abbreviation: mAP: Mean average precision.

methods, multiple methods of eliminating mismatch points, and weighted fusion to achieve real-time image stitching. A total of 51 images were stored, including the input images, feature point matching results, and real-time stitched images. Data such as feature extraction time, the number of feature points, the number of feature point matches, and the total processing time were recorded in a Microsoft Excel file for documentation.

5.2. Real-time image stitching

In this study, two hexacopters and one octocopter were employed (**Figure 25**). The UAV webcams capture the real-time images; when three UAVs operate simultaneously at the same altitude and along the same path, each UAV captures an image every 5 s and transmits it to the ground station for real-time image stitching and pollution identification. The images are stitched directly, without translation or rotation, during the image stitching process.

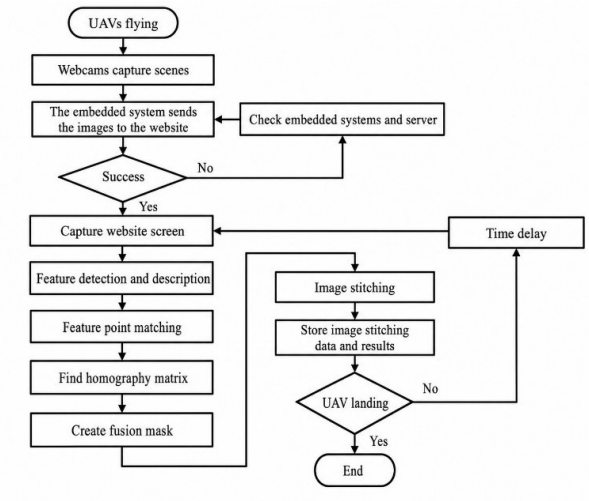

Figure 24. The real-time image-stitching process

In this study, the three input images for image stitching were captured by UAVs, as shown in **Figure 26**. Comparisons of image stitching using SIFT, ORB, and AKAZE were conducted

for feature detection and description. Results without weighted fusion were compared with those obtained with weighted fusion. The image stitching results using ORB without weighted fusion are shown in **Figure 27**, whereas the image stitching result using weighted fusion is shown in **Figure 28**. **Table 4** presents the computing time using different stitching combinations without weighted fusion, whereas **Table 5** presents the results for weighted fusion.

Figure 25. The unmanned aerial vehicles used in the experiment

Table 4. The total time without weighted fusion using different methods

Method	SIFT	ORB	AKAZE
BF+RANSAC time (s)	6.42	1.75	5.18
BF+PROSAC time (s)	6.68	2.17	4.79
FLANN+RANSAC time (s)	5.09	1.12	4.67
FLANN+PROSAC time (s)	4.82	1.09	4.70

Abbreviations: AKAZE: Accelerated KAZE; BF: Brute-force; FLANN: Fast Library for Approximate Nearest Neighbors; ORB: Oriented Features from Accelerated Segment Test and Rotated Binary Robust Independent Elementary Features; PROSAC: Progressive sample consensus; RANSAC: Random sample consensus; SIFT: Scale-Invariant Feature Transform.

An image-stitching comparison was conducted to demonstrate the effectiveness of weighted fusion in eliminating seams. Using the same input images and various stitching methods, we found that weighted fusion (**Figure 28**) effectively eliminated seams and corrected color discrepancies. Seam issues were prominent without weighted fusion, as shown in **Figure 27**. We combined multiple feature-point-matching methods for feature detection, description, and image stitching. The best-to-worst results were SIFT, AKAZE, and ORB. ORB showed the most pronounced misalignment, AKAZE had a slight misalignment, and SIFT performed best. **Tables 4** and **5** show that, in terms of total computation time, the order from fastest to slowest is ORB, AKAZE, and SIFT.

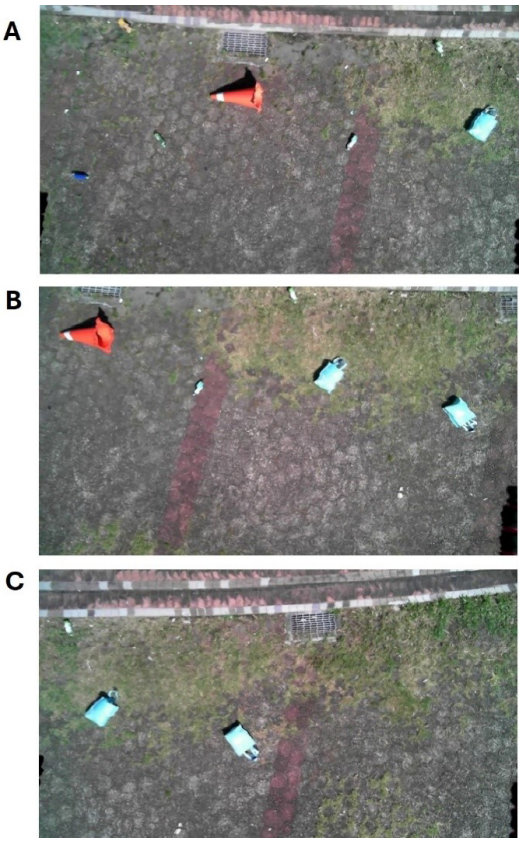

Figure 26. Original images from the unmanned aerial vehicle: (A) Left, (B) middle, and (C) right-view images.

Table 5. The total time with weighted fusion using different methods

Method	SIFT	ORB	AKAZE
BF+RANSAC time (s)	7.70	3.10	5.65
BF+PROSAC time (s)	7.79	3.20	6.60
FLANN+RANSAC time (s)	6.81	2.59	5.71
FLANN+PROSAC time (s)	6.43	2.37	5.54

Abbreviations: AKAZE: Accelerated KAZE; BF: Brute-force; FLANN: Fast Library for Approximate Nearest Neighbors; ORB: Oriented Features from Accelerated Segment Test and Rotated Binary Robust Independent Elementary Features; PROSAC: Progressive sample consensus; RANSAC: Random sample consensus; SIFT: Scale-Invariant Feature Transform.

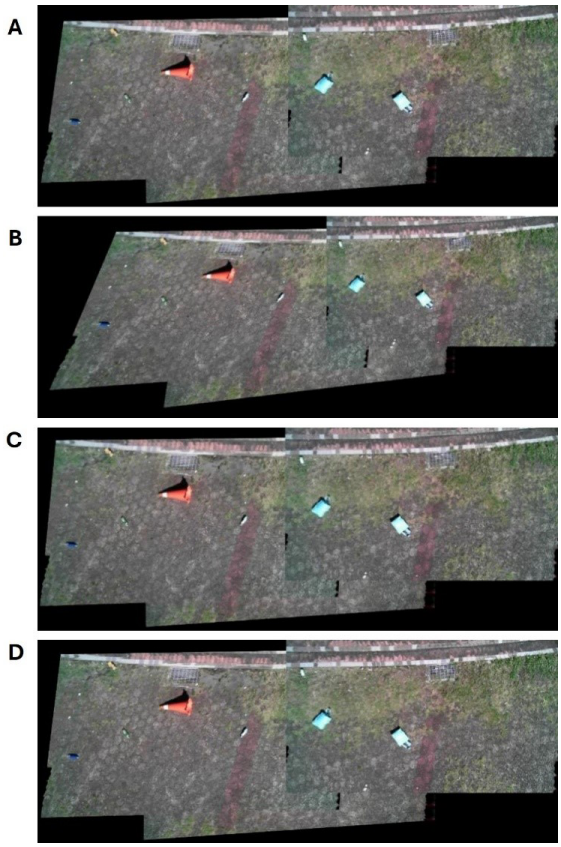

Figure 27. Image stitching using ORB without weighted fusion: (A) BF+RANSAC, (B) BF+PROSAC, (C) FLANN+RANSAC, and (D) FLANN+PROSAC.
Abbreviations: BF: Brute-force; FLANN: Fast Library for Approximate Nearest Neighbors; ORB: Oriented Features from Accelerated Segment Test and Rotated Binary Robust Independent Elementary Features; PROSAC: Progressive sample consensus; RANSAC: Random sample consensus.

Figure 28. Image stitching using ORB+FLANN+PROSAC with weighted fusion.
Abbreviations: FLANN: Fast Library for Approximate Nearest Neighbors; ORB: Oriented Features from Accelerated Segment Test and Rotated Binary Robust Independent Elementary Features; PROSAC: Progressive sample consensus.

5.3. Real-time pollution identification

Since the real-time UAV-based riverine waste monitoring system uses YOLO in the Darknet framework and cannot use YOLO in the PyTorch framework, it was modified to adopt the Darknet framework for training YOLOv5. The models included YOLOv5 variants (*tiny*, *s*, *m*, *l*, *x*), as well as the original v5 model. The training set was divided into three classes: bottle, garbage, and negative samples (rock). The training iterations were set to 20,000, and the training dataset consisted of 2,620 images. The average loss quickly converges, with the mAP stabilizing at 69%. The average loss begins to converge at 16,000 training iterations, and the mAP stabilizes at 70%. Based on these results, no further increase in the number of training iterations is needed for most models.

The UAV-based riverine waste monitoring system was used to detect pollution and obtain streaming data via Kafka. The data included area size, current coordinates, and waste type. At the same time, real-time images were transmitted via imageZMQ, and the results were sent to the monitoring terminal. In **Figure 29**, the UAV-based riverine waste monitoring system at National Taiwan Ocean University identifies pollution, transmits coordinates and pollution-area information to the monitoring terminal, and then generates a heat map, as shown in **Figure 30**.

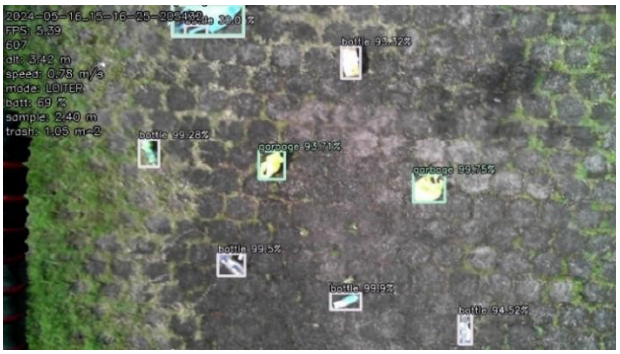

Figure 29. Pollution identification

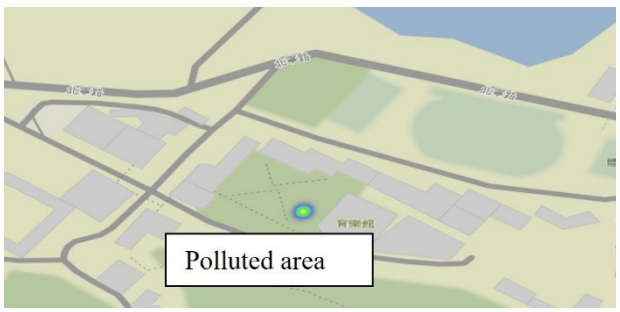

Figure 30. The monitoring terminal shows a pollution heat map

5.4. Identification after stitching

The results employ the Darknet version of YOLOv5, the PyTorch version of YOLOv8, and instance segmentation for waste identification, as shown in **Figures 31–33**. Most waste was successfully identified, and the confidence scores were relatively high. To investigate which YOLO method has a higher recognition success rate, a comparison of recognition and correct rates was conducted, as presented in **Tables 6–8**. In YOLOv5, model-l has the highest recognition and correct rates. In YOLOv8, model-n and model-x achieved 100% accuracy, but both recognition rates are low. Model-m has a better trade-off between recognition and correct rates. In YOLOv8-segment, model-l has outstanding performance in recognition and correct rates, both at 100%. This is because the YOLOv8-segment model-l produces pixel-level mask outputs, providing precise object shape information and better object-overlap separation than other models. The recognition rate is defined as the total number of waste items (garbage and bottles) in the image, divided by the number of successfully recognized items (garbage and bottles). For example, if there are eight waste items in the image, the denominator is 8, and the numerator is the number of items successfully recognized. The correct rate is defined as the number of correctly identified items (garbage and bottles) divided by the number of successfully recognized items (garbage and bottles).

The results indicate that YOLOv8 instance segmentation performed the best, as shown in **Figure 33**. It accurately identifies the bottles and garbage in the image, although some misidentifications still occur. Since YOLOv8l-seg achieves 100% recognition and accuracy rates, this model was applied to our garbage identification task. The least effective was YOLOv5, which not only failed to achieve a 100% recognition rate but also showed relatively low accuracy.

Figure 31. The YOLOv5l identification result

Figure 32. The YOLOv8l identification result

Figure 33. The YOLOv8l-seg identification result

Table 6. YOLOv5 waste identification results

Model	***tiny***	***s***	**v5**	***m***	***l***	***x***
Recognition rate	75%	87.5%	62.5%	75%	87.5%	75%
Correct rate	66.7%	71.4%	60%	83.3%	85.7%	83.3%

Table 7. YOLOv8 waste identification results

Model	***n***	***s***	***m***	***l***	***x***
Recognition rate	50%	100%	87.5%	75%	75%
Correct rate	100%	75%	85.7%	83.3%	100%

Table 8. YOLOv8-seg waste identification results

Model	***n***	***s***	***m***	***l***	***x***
Recognition rate	50%	87.5%	87.5%	100%	87.5%
Correct rate	75%	71.4%	100%	100%	100%

6. Conclusion and future prospects

This study proposes a multi-UAV system based on the YOLO deep learning neural network to rapidly and effectively detect waste in riverine

and riverbank areas. Optimal stitching of images and reduced computational time can be achieved using ORB, FLANN, and PROSAC with weighted fusion. The system can identify the size of the waste area, the type of waste, and the current coordinates from stitched images and transmit the images to the ground control center. Real-time image-stitching experiments were successfully conducted on campus using two hexacopters and one octocopter. Two image-stitching datasets were tested, and the proposed method with weighted fusion preserves 100% of the content of the original images.

Different models of YOLOv5 and YOLOv8 networks were used to detect waste pollution. An optimal model of YOLOv8 that can identify waste items with a 100% recognition rate and 100% accuracy rate was obtained. A novel method that uses GPS data to reduce computation time was presented, which divides the captured images into subgroups based on their GPS locations to reduce computational cost during stitching.[39] It was achieved by reducing the number of matches to be considered compared with traditional approaches. The concept of reducing matches was also used in our study. Compared with conventional SIFT-based stitching, the proposed stitching process can reduce computation time by more than 60%. In this study, three images from three UAV cameras are stitched in 2.37 seconds, which is sufficient for the garbage detection task. Three images are matched in two steps; the number of matches in the second step is reduced because ORB produces fewer matched points and runs faster.

Analysis of the experimental process and results identified the main factors affecting stitching outcomes as follows:

(i) Vibration caused by the UAVs.
(ii) Blurry images due to the inability of webcams to autofocus when UAVs move too fast or shake violently.
(iii) Failure to maintain a fixed direction and angle in high wind speeds.
(iv) Considerable distance variations between the three UAVs, which can lead to stitching failures.
(v) If there is a disparity in the distance between three real-time transmitted images, weighted fusion may result in the deletion of some information.

To address these issues, dual GNSS with RTK technology and improved flight control systems were employed to enhance UAV stability and positioning accuracy.

In the future, YOLO's recognition effectiveness and variability can be enhanced by incorporating more diverse training datasets or different types of waste training sets. The YOLO model's recognition capabilities and variability can be improved by incorporating more varied training datasets or different types of waste training sets. These diverse datasets should include waste images across various environments, lighting conditions, and seasons. The real-time UAV-based riverine waste monitoring system can be enhanced using YOLO with the PyTorch framework for waste detection, or other methods, to achieve real-time UAV waste detection. For real-time image stitching, optimizing image stitching performance can be achieved using more effective techniques.

In this study, illumination-smoothing image stitching[40] was not considered. Our tests show that the UAV captures nearby images almost simultaneously, with nearly identical lighting for image stitching. Thus, the light inconsistency was not considered in this study. The lighting issue will be addressed in our future work. A study presented a model that outperformed existing YOLO models for detecting moving objects, particularly achieving higher accuracy for small objects (8×8 pixels).[41]

To improve small-object detection, data augmentation techniques were employed. The model was then fine-tuned using transfer learning with the new dataset. This method can be applied to small object detection in our future work. Using higher-quality photographic equipment and positioning systems can also reduce noise in the UAV's returned images, further improving image-stitching quality. In addition, energy harvesting technologies can enhance system endurance and stability.[42] Incorporating such innovations may further support long-term, real-time waste monitoring missions.

Acknowledgments

None.

Funding

National Science and Technology Council, Taiwan (NSTC 112-2221-E-019 -040-MY2).

Conflict of interest

The authors declare they have no competing

interests.

Author contributions

Conceptualization: Jih-Gau Juang
Investigation: All authors
Methodology: All authors
Formal analysis: Cheng-Chien Su
Writing – original draft: Cheng-Chien Su
Writing – review & editing: Jih-Gau Juang

Availability of data

Data are obtained from field tests and measurements.

AI tools statement

All authors confirm that no AI tools were used in the preparation of this manuscript.

References

1. Liao YH, Juang JG. Automatic marine debris inspection. *Aerospace.* 2023;10(1):84.

   https://doi.org/10.3390/aerospace10010084

2. Liao S. HAIDA-trash-dataset-high-resolution-aerial-image. Published 2021. Accessed April 25, 2025. https://github.com/LiaoSteve/HAIDA-Trash-Dataset-High-Resolution-Aerial-image

3. He X, He L, Li X. Image stitching via convolutional neural network. In: *2021 7th International Conference on Computer and Communications (ICCC).* Piscataway, NJ: IEEE; 2021:709-713.

   https://doi.org/10.1109/ICCC54389.2021.9674411

4. Kumareswaran D, Nasir N, Rahman MZA, *et al.* Computational speed and qualitative assessment of real-time image stitching algorithm. In: *2021 International Conference on Communication, Control and Information Sciences (ICCISc).* Piscataway, NJ: IEEE; 2021:1-6.

   https://doi.org/10.1109/ICCISc52257.2021.9484887

5. Lowe DG. Distinctive image features from scale-invariant keypoints. *Int J Comput Vis.* 2004;60(2):91-110.

   https://doi.org/10.1023/B:VISI.0000029664.99615.94

6. Rublee E, Rabaud V, Konolige K, *et al.* ORB: an efficient alternative to SIFT or SURF. In: *2011 International Conference on Computer Vision.* Piscataway, NJ: IEEE; 2011:2564-2571.

   https://doi.org/10.1109/ICCV.2011.6126544

7. Alcantarilla P, Nuevo J, Bartoli A. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In: *Proceedings of the British Machine Vision Conference 2013.* Durham, UK: British Machine Vision Association; 2013:13.1-13.11.

   https://doi.org/10.5244/C.27.13

8. Jakubović S, Velagic J. Image feature matching and object detection using brute-force matchers. In: *2018 International Symposium ELMAR.* Piscataway, NJ: IEEE; 2018:83-86.

   https://doi.org/10.23919/ELMAR.2018.8534641

9. OpenCV. Basics of brute-force matcher. Accessed June 15, 2025. https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html

10. Yang Z, Qin X. Image stitching technology based on SIFT, FLANN, and RPOSAC algorithms. In: *2023 4th International Conference on Computer Vision, Image and Deep Learning (CVIDL).* Piscataway, NJ: IEEE; 2023:52-56.

    https://doi.org/10.1109/CVIDL58838.2023.10167270

11. Muja M, Lowe DG. FLANN-fast library for approximate nearest neighbors. In: *Proceedings of the Fourth International Conference on Computer Vision Theory and Applications.* Setúbal, Portugal: SciTePress - Science and Technology Publications; 2009;2:331-340. Accessed April 20, 2025. https://www.scienceopen.com/document?vid=a6407082-f90a-4b9b-aa76-0b7796497bcd

12. Zhang Y, Zhang J, Zhang L, *et al.* Research on panorama reconstruction technique of UAV aerial image based on improved ORB algorithm. In: *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE).* Piscataway, NJ: IEEE; 2019:1252-1256.

    https://doi.org/10.1109/EITCE47263.2019.9095113

13. OpenCV. Understanding k-nearest neighbors. Accessed January 11, 2025. https://docs.opencv.org/4.x/d5/d26/tutorial_py_knn_understanding.html

14. Fischler MA, Bolles RC. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Readings in Computer Vision.* Amsterdam, Netherlands: Elsevier; 1987:726-740.

    https://doi.org/10.1016/B978-0-08-051581-6.50070-2

15. Gan W, Wu Z, Wang M, Cui X. Image stitching based on optimized SIFT algorithm. In: *Proceedings of 2023 5th International Conference on Intelligent Control, Measurement and Signal Processing (ICMSP).* Piscataway, NJ: IEEE; 2023:1099-1102.

https://doi.org/10.1109/ICMSP58539.2023.10170989

16. Chum O, Matas J. Matching with PROSAC - progressive sample consensus. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Piscataway, NJ: IEEE; 2005;1:220-226.

    https://doi.org/10.1109/CVPR.2005.221

17. Jeon HK, Jeong JM, Lee KY. An implementation of the real-time panoramic image stitching using ORB and PROSAC. In: *2015 International SoC Design Conference (ISOCC)*. Piscataway, NJ: IEEE; 2015:91-92.

    https://doi.org/10.1109/ISOCC.2015.7401661

18. Li H, Wang J, Han C. Image mosaic and hybrid fusion algorithm based on pyramid decomposition. In: *2020 International Conference on Virtual Reality and Visualization (ICVRV)*. Piscataway, NJ: IEEE; 2020:205-208.

    https://doi.org/10.1109/ICVRV51359.2020.00049

19. Xiu C, Ma Y. Image stitching based on improved gradual fusion algorithm. In: *2019 Chinese Control and Decision Conference (CCDC)*. Piscataway, NJ: IEEE; 2019:2933-2937.

    https://doi.org/10.1109/CCDC.2019.8832814

20. Wang SI. Title in Pinyin Chinese [Yingyong Dichengben Shuangpin GNSS RTK Jishu Yu Wurenji Dingwei Dingxiang Zhi Yanjiu]. *J Photogramm Remote Sens.* 2020;25(4):241-252. [In Chinese]

    https://doi.org/10.6574/JPRS.202012_25(4).0004

21. Jocher G, Chaurasia A, Qiu J. Ultralytics YOLO (Version 8.0.0) [computer software]. Accessed August 27, 2024. https://github.com/ultralytics/ultralytics

22. Bawankule R, Gaikwad V, Kulkarni I, *et al.* Visual detection of waste using YOLOv8. In: *2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS)*. Piscataway, NJ: IEEE; 2023:869-873.

    https://doi.org/10.1109/ICSCSS57650.2023.10169688

23. Sapkota R, Ahmed D, Karkee M. Comparing YOLOv8 and Mask R-CNN for instance segmentation in complex orchard environments. *Artif Intell Agric.* 2024;13:84-99.

    https://doi.org/10.1016/j.aiia.2024.07.001

24. Logitech BRIO ULTRA HD PRO webcam. Accessed August 2, 2024. https://www.logitech.com/zh-tw/products/webcams/brio-4k-hdr-webcam.960-001105.html

25. Bass J. imagezmq: A Set of Python Classes That Transport OpenCV Images from One Computer to Another Using PyZMQ Messaging. Accessed March 12, 2025. https://github.com/jeffbass/imagezmq#why-use-imagezmq

26. Gimbal Overview. ArduPilot. Accessed March 2, 2025. https://ardupilot.org/copter/docs/common-tarot-gimbal.html

27. Rosten E, Drummond T. Machine learning for high-speed corner detection. In: *Lecture Notes in Computer Science*. Berlin, Germany: Springer; 2006:430-443.

    https://doi.org/10.1007/11744023_34

28. Calonder M, Lepetit V, Strecha C, *et al.* BRIEF: binary robust independent elementary features. In: *Lecture Notes in Computer Science*. Berlin, Germany: Springer; 2010:778-792.

    https://doi.org/10.1007/978-3-642-15561-1_56

29. Homography matrix transformation. Accessed September 12, 2024. https://lxrd-aj.github.io/blog/cv/homography/

30. Chiang CH, Juang JG. Application of UAVs and image processing for riverbank inspection. *Machines.* 2023;11(9):876.

    https://doi.org/10.3390/machines11090876

31. Ultralytics YOLOv5 architecture. Accessed March 2, 2025. https://docs.ultralytics.com/yolov5/tutorials/architecture_description/#41-compute-losses

32. YOLOv5 architecture description. *arXiv.* 2024. Accessed March 2, 2025. https://arxiv.org/html/2407.20892v1

33. What is YOLOv8? Roboflow Blog. Accessed May 2, 2025. https://blog.roboflow.com/whats-new-in-yolov8/

34. YOLOv8 architecture description. Medium. Accessed May 2, 2025. https://henry870603.medium.com/object-detection-yolov8%E8%A9%B3%E8%A7%A3-fdf8874e5e99

35. About the functionality of DFL_loss. GitHub Issues. Accessed May 2, 2025. https://github.com/ultralytics/ultralytics/issues/4219?trk=article-ssr-frontend-pulse_little-text-block

36. YOLO8 loss function for classification task. GitHub Issues. Accessed May 2, 2025. https://github.com/ultralytics/ultralytics/issues/4684?trk=article-ssr-frontend-pulse_little-text-block

37. Feng C, Zhong Y, Gao Y, *et al.* TOOD: task-aligned one-stage object detection. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Piscataway, NJ: IEEE; 2021:3490-3499.

    https://doi.org/10.1109/ICCV48922.2021.00349

38. What is instance segmentation? Roboflow Blog. Accessed October 23, 2025. https://blog.roboflow.

com/instance-segmentation-roboflow/

39. Perera CJ, Premachandra C, Kawanaka H. Low pixel resolution hyperspectral image mosaics generation using learning-based feature matching. *IEEE Access.* 2023;11:104084-104093.

    https://doi.org/10.1109/ACCESS.2023.3315769

40. Liu S, Chai Q. Shape-optimizing and illumination-smoothing image stitching. *IEEE Trans Multimedia.* 2019;21(3):690-703.

    https://doi.org/10.1109/TMM.2018.2864576

41. Nakaoka Y, Asaki S, Premachandra C. E-YOLO to OMOD: an enhanced YOLO framework for small moving object detection in omnidirectional videos. *IEEE Open J Intell Trans Syst.* 2026;7:838-846.

    https://doi.org/10.1109/OJITS.2026.3670217

42. Citroni R, Di Paolo F, Livreri P. A novel energy harvester for powering small UAVs: performance analysis, model validation and flight results. *Sensors.* 2019;19(8):1771.

    https://doi.org/10.3390/s19081771

Cheng-Chien Su *received the M.S. degree from the Department of Communications, Navigation and Control Engineering, National Taiwan Ocean University, Keelung, Taiwan. His research interests include image processing and image analysis. He is currently a hardware engineer with Wincomm Corporation, Hsinchu, Taiwan.*

Jih-Gau Juang r*eceived the Ph.D. degree from the University of Missouri-Columbia, Missouri, USA. He is currently a Distinguished Professor with the Department of Communications, Navigation and Control Engineering, National Taiwan Ocean University, Keelung, Taiwan. His research interests include intelligent systems, vehicle control, adaptive control, and intelligent control.*

https://orcid.org/0000-0003-2683-9931

An International Journal of Optimization and Control: Theories & Applications (https://accscience.com/journal/ijocta)