# A Related Works

**HSI-based sea surface effect mitigation** In [42], an unsupervised classification method is proposed to mitigate sea surface effects by employing an independent component analysis (ICA) mixture model. Similarly, the authors of [46, 57, 13] have studied ICA, minimum noise fraction (MNF) transform, and principal component analysis (PCA) methods for sea surface effect mitigation. The above works have shown HSIs can remove sea surface effects; however, the HSI sensor data have a low resolution for object detection in far-field aerial data.

**Uni-modal object detection** For sea surface maritime object detection, the authors of [33, 35] have conducted benchmarks for various object detection methods, including R-CNN [17], Fast R-CNN [16], Faster R-CNN [40], SSD [29], YOLO [39], and RetinaNet [27]. For maritime object detection in aerial RGB data, the FGSD [7] dataset provides bounding-box-annotated RGB data. In [48, 28], a light and storage-efficient maritime object detection method has been proposed, in which the loss function and size of anchor boxes are modified based on the YOLO [39]. For object detection in aerial HSI data, a CNN-based maritime surveillance system is proposed [14], in which a pixel-wise classification method is leveraged instead of a bounding-box-annotated dataset. Similarly, [53] have proposed a two-stage deep learning method for object detection, in which two CNNs (pixel-wise and region-based) are used to extract spectral and spatial features, respectively. Bounding-box-annotated object detection methods are timely and necessary for maritime surveillance systems; however, the existing bounding-box-annotated dataset [52] only includes close-up terrestrial HSI data instead of aerial maritime data.

**Multi-modal object detection** Various types of sensors (*e.g.*, RGB, radar, HSI, etc.) have been widely used for multi-modal maritime surveillance systems. Fusing multi-sensor information and making accurate inferences have drawn much attention [12]. In [5], a multi-modal system is proposed for joint decision-making based on information fusion, in which various sensors such as radar, pan-tilt-zoom (PTZ) cameras, and automatic-identification-systems (AIS) cameras are fused. In [30], a challenge result is presented for object detection tasks on the multi-modal dataset, UNICORN V2, which includes synthetic aperture radar (SAR) and optical sensor data. Both HSI and SAR data can be utilized as complementary options for optical sensors; however, in maritime domains, HSI data is more apt for complementation since it captures information about a material's intrinsic properties, whereas SAR data provides structural information about a scene [22]. For maritime object detection, the authors of [18, 9] have proposed a robust real-time object detection method in sea surface images. Although various multi-modal methods have been studied, multi-modal object detection methods for maritime object detection in aerial RGB/HSI data have not been studied.

# B Data Collection Method

As described in the main paper, our focus is on a new dataset, '**M$^2$SODAI**,' for maritime object detection with synchronized RGB and HSI data. Here, we thoroughly introduce our data collection method to answer the following two questions: 1) "*where did we acquire the data?*" and 2) "*how did we acquire the data?*."

**Where did we acquire the data?** We did 59 flight strips in 12 flight measurement campaigns to collect aerial data. For the diversity of the collected data, we had flights in 11 different spots during the data acquisition. Table B.1 demonstrates the history of our data acquisition, in which the location, latitude, longitude, coverage area, date, and the number of collected data are shown. As shown in the table, our flight measurement campaigns cover a total area of 299.7 km$^2$. Also, after our data processing procedure, we obtain 1,257 pairs of aerial RGB and HSI data. Figure B.1 visualizes the collected raw data. In the first row, we depict the raw RGB data, which have a significantly higher resolution than the HSI data. In the second row, merged visualizations of the HSI data in flight strips are depicted, in which the HSI data are transformed into RGB images for visualization. In the third row, the raw HSI data obtained in each flight strip are depicted.

**How did we acquire the data?** For the aerial data acquisition system, we used a single-engine utility aircraft equipped with an RGB sensor and an HSI sensor. During the flights, the aircraft

Table B.1: Details of the locations of collected data. The name, latitude, and longitude of the locations are provided. In addition, we write the number of the data including target objects, after data processing for each of the eleven different locations.

| **No.** | **Location** | **Latitude (degree)** | **Longitude (degree)** | **Coverage Area** | **Number of Data w/ Target Objects** |
|---|---|---|---|---|---|
| 1 | Dongbaek Bridge, Gunsan City | N35.9984° | E126.7047° | $5.36 \times 3.73$ km$^2$ | 176 |
| 2 | Saemangeum Seawall Bridge, Gunsan City | N35.9029° | E126.5076° | $7.32 \times 8.04$ km$^2$ | 25 |
| 3 | Mokpo North Port, Mokpo City | N34.8006° | E126.3529° | $5.24 \times 4.07$ km$^2$ | 144 |
| 4 | Port of Mokpo, Mokpo City | N34.7817° | E126.3835° | $3.74 \times 3.74$ km$^2$ | 10 |
| 5 | Port of Mukho, Donghae City | N37.5489° | E129.1222° | $5.41 \times 6.86$ km$^2$ | 42 |
| 6 | Odori Beach, Pohang City | N36.1554° | E129.3974° | $6.20 \times 4.58$ km$^2$ | 78 |
| 7 | Yeongilman Port, Pohang City | N36.1060° | E129.4254° | $3.86 \times 1.87$ km$^2$ | 24 |
| 8 | Pohang Ferry Terminal, Pohang City | N36.0541° | E129.3800° | $3.65 \times 4.43$ km$^2$ | 56 |
| 9 | Jeongok Port, Hwaseong City | N37.1886° | E126.6509° | $5.92 \times 4.28$ km$^2$ | 83 |
| 10 | Gungpyeong Port, Hwaseong City | N37.1163° | E126.6813° | $3.12 \times 1.92$ km$^2$ | 53 |
| 11 | Guk-Dong Port #1, Yeosu City | N34.7151° | E127.7256° | $5.63 \times 5.97$ km$^2$ | 287 |
| 12 | Guk-Dong Port #2, Yeosu City | N34.7151° | E127.7256° | $5.45 \times 5.83$ km$^2$ | 279 |
|  |  |  |  | **Total** | 1257 |

maintains its speed of 260 km/h and altitude of 1 km. The collected RGB data are saved as `‘.tiff’` filename extension, and the HSI data are saved as `‘.hdr’` and `‘.bsq’` filename extensions, in which `‘.hdr’` and `‘.bsq’` files contain specifications and reflectance information of the HSI data, respectively.

## C Data Usage Guideline

**Guideline** The M$^2$SODAI dataset is a multi-modal dataset that provides RGB sensor images and hyperspectral sensor images for maritime object detection. The dataset can be downloaded through the link presented later and consists of three folders (train, val, test) as depicted in Fig. C.1. Each `‘.jpg’` extension file contains the RGB sensor image, and the `‘.json’` extension file contains the ground truth label annotations, where the size of the image is $1600 \times 1600 \times 3$. The label file conforms to the COCO format and displays the rectangular bounding boxes of objects. In addition, the `‘.mat’` extension file with the name matching each RGB sensor image contains hyperspectral image information ($224 \times 224 \times 127$). For ease of use, we will share a tutorial on loading our dataset from the popular object detection open-source library via GitHub code.

**Maintenance** The M$^2$SODAI dataset is ever-lastingly available on our homepage and will be maintained by Jonggyu Jang (jgjang@postech.ac.kr, jgjang0123@gmail.com, https://jonggyu.me) Furthermore, the dataset is licensed under MIT License and will be updated if any issues will be handled through GitHub Issue tab.

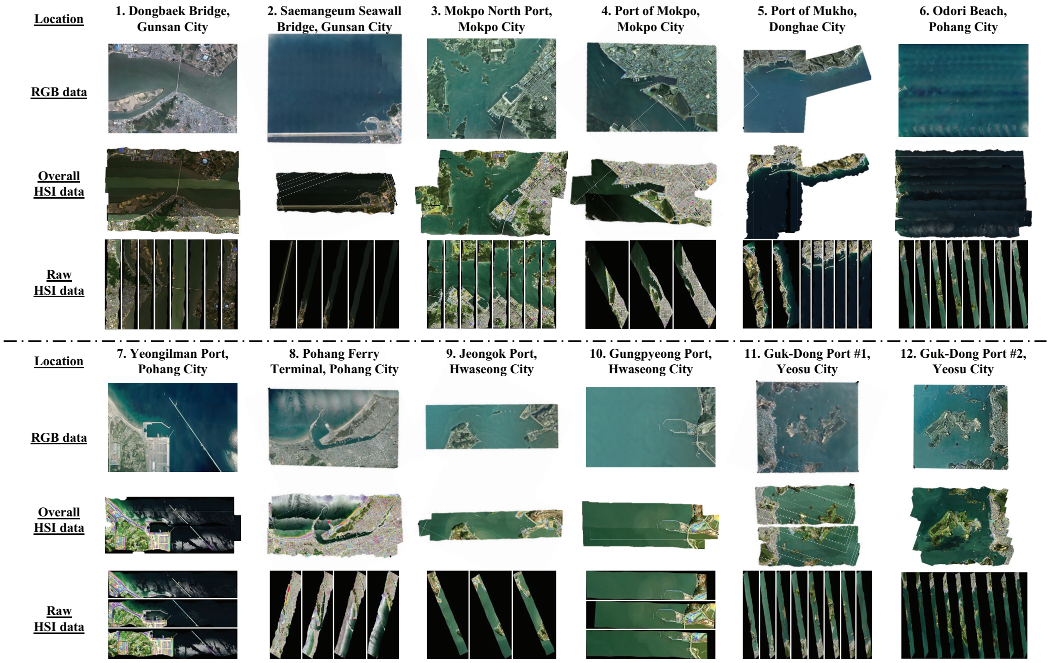

Location
1. Dongbaek Bridge, Gunsan City
2. Saemangeum Seawall Bridge, Gunsan City
3. Mokpo North Port, Mokpo City
4. Port of Mokpo, Mokpo City
5. Port of Mukho, Donghae City
6. Odori Beach, Pohang City
RGB data
Overall HSI data
Raw HSI data
Location
7. Yeongilman Port, Pohang City
8. Pohang Ferry Terminal, Pohang City
9. Jeongok Port, Hwaseong City
10. Gungpyeong Port, Hwaseong City
11. Guk-Dong Port #1, Yeosu City
12. Guk-Dong Port #2, Yeosu City
RGB data
Overall HSI data
Raw HSI data

Figure B.1: Illustration of the collected raw data. We collected the data on twelve spots. The first row shows the collected raw RGB data. The second and third rows show the overall HSI data and collected raw HSI data in each flight strip. In this figure, since the sensors have different specifications on the field of view (FoV), the raw RGB data and HSI data have different coordinates.

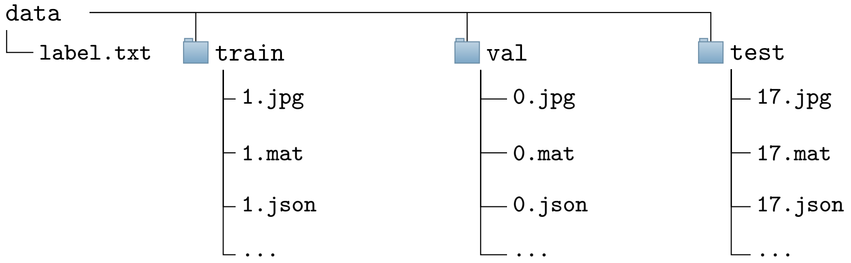


Figure C.1: Dataset Structure

## D Image Registration

In this section, we thoroughly introduce our image registration procedure. Since the RGB sensor has a wider field of view (FoV) specification than the HSI sensor, the collected RGB data covers a broader area than the HSI data. Hence, the pixel locations in the RGB and HSI data are not precisely matched, as depicted in Fig. B.1. To synchronize the coordinates of RGB and HSI data, we conducted an image registration method into the collected data. However, if we convert the RGB and HSI data into gray-scale images as in standard image registration methods, there are few matched features between the data. Thus, the data cannot be successfully registered in most cases. To remedy the problem, we use a contrast enhancer that enables us to discover additional features in the gray-scale data. Then, we apply the existing feature extractor/matcher to register the data.

**Visualization of RGB data [24]** The raw RGB images have a huge file size due to their high resolution, making them difficult to handle with the image-matching function of the OpenCV library [4]. Hence, for the sake of tractable size, we down-scaled the raw RGB data into 1/7 scale, in which the down-scaled RGB data have a similar resolution to the raw HSI data. Then, we converted them into gray-scale images.

**Visualization of HSI data [24]** For image registration, it is general to convert HSI data to gray-scale images. First, we converted the raw HSI data into RGB-format images by treating the following spectral channels shown below as R, G, and B channels: i) Red channel: 53rd channel of HSI data (wavelength of 626 nm), ii) Green channel: 34th channel of HSI data (wavelength of 538 nm), and iii) Blue channel: 17th channel of HSI data (wavelength of 459 nm). Second, we transformed the RGB-format HSI images into gray-scale images.

**Contrast enhancer** Before the image registration, we applied a contrast enhancer to the converted RGB and HSI data to make them have rich features. The source code of the contrast enhancer is provided in Listing 1. First, the input image is converted to the LAB color model, in which the $L$ component represents the brightness of the image, the $a^*$ channel represents the relative green-red opponent, and the $b^*$ channel represents the relative yellow-blue opponent. Second, the contrast of the input image is enhanced by applying contrast-limited adaptive histogram equalization (CLAHE). Finally, contrast-enhanced images can be obtained by merging the enhanced $L$ component, $a^*$ component, and $b^*$ component.

```python
import cv2
def img_Contrast(img):
    # 1. Converting image to LAB Color model
    LAB = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    # 2. Splitting the LAB image
    L, A, B = cv2.split(LAB)
        # L : brightness of the image
        # A : relative green-red color
        # B : relative yello-blue color
    # 3. Applying CLAHE histogram equalizer
    CLAHE = cv2.createCLAHE(clipLimit=5.0, tileGridSize=(16, 16))
    EL = CLAHE.apply(L)
    # 4. Merge channels and convert them to RGB
    ELAB = cv2.merge((EL, A, B))
    return cv2.cvtColor(ELAB, cv2.COLOR_LAB2BGR)
```

Listing 1: Source code of the contrast enhancer.

**Feature extractor and matcher** The image registration procedure consists of 1) feature extraction and 2) feature matcher. The source code of the feature detector and matcher is provided in Listing 2. In the source code, `‘img1’` and `‘img2’` denote the source image (RGB data) and destination image (HSI data), the contrasts of which are enhanced by the contrast enhancer in Listing 1. In the first step, we define an oriented FAST and rotated BRIEF (ORB) detector, which is a combination of the FAST keypoint detector and the BRIEF descriptor. In the second step, by applying the ORB detector to the input images, we have keypoints (features) and descriptors (feature matching information) of the images. In the third step, the matched features between two data (`‘img1’` and `‘img2’`) are obtained by a Brute-force matcher. In the feature matcher, we use norm hamming metric since the descriptors of the ORB detector are binary vectors. Finally, we obtain the homography matrix between the RGB and HSI data by solving the least-square problem of the matched keypoints. By multiplying the homography matrix with the pixel indices of the RGB data, we can obtain the corresponding pixel indices in the HSI data.

```python
import cv2
import numpy as np
def get_match(img1, img2):
    # 1. Define ORB feature detector
    orb=cv2.ORB_create(
        nfeatures=100000,
    )
    # 2. Detect features of HSI and RGB
    kpA, desA = orb.detectAndCompute(img1,None)
    kpB, desB = orb.detectAndCompute(img2,None)
    # 3. Brute-Force feature matcher
    bf =cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
    matches=bf.match(desA,desB)
    # 4. obtain homography matrix
    src = np.float32([kpA[m.queryIdx].pt for m in
   matches]).reshape((-1,2))
    dst = np.float32([kpB[m.trainIdx].pt for m in
   matches]).reshape((-1,2))
    H, _ = cv2.findHomography(src, dst, cv2.RANSAC)
    return H
```

Listing 2: Source code of the feature extractor, feature matcher, and homography matrix finder.

**Offset correction** In order to ensure the reliability of the dataset, we carefully double-checked the pixel offsets between the registered HSI and RGB data. Although most of the data are perfectly matched, some data have small pixel offsets; thus, we manually corrected the pixel offsets.

In Fig. D.1, we illustrate 15 annotated data randomly sampled from our training dataset.

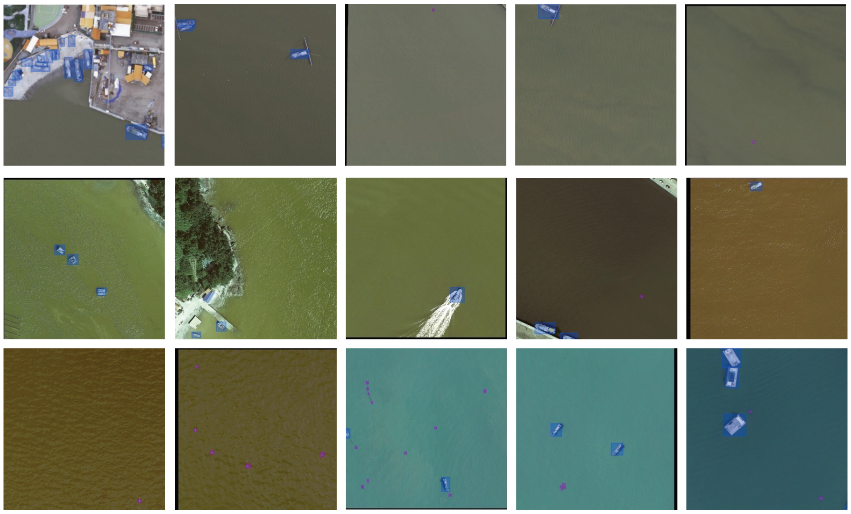
## E Further Dataset Analysis

In this section, we analyze the $M^2$SODAI dataset in detail. For the analysis, we sample several cropped data among the training data, which are categorized into four categories: i) ships, ii) floating matters, iii) background with sea surface effects, and iv) background with the clean sea surface. Figure E.1a shows the visualized RGB data and the reflectance patterns of the corresponding HSI data. As shown in the figure, ships appear to be strongly distinct from the other categories in both reflectance patterns and RGB data. On the other hand, the cropped RGB data of floating matters and backgrounds with sea surface effects are similar to each other, which can cause confusion for RGB-based detection models. However, because the reflectance patterns of the two categories are significantly different, the HSI data can distinguish the two categories. By comparing the backgrounds with and without the sea surface effects, they have similar reflection patterns because their intrinsic reflection properties are almost the same.

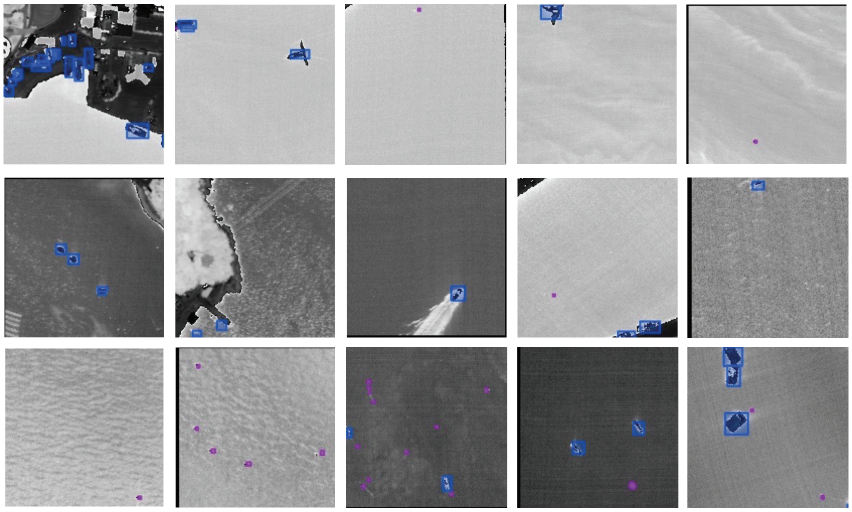
For a more detailed analysis, we compare the similarity of cropped data of target objects and backgrounds. In Fig. E.1b, we depict a table to represent the category-wise similarity in RGB data. The similarity metric measures the cosine similarity score of the embedding vectors of images. Specifically, we obtain an encoded vector of the input image using the contrastive language-image pre-training (CLIP) model [37], *i.e.*, the image is transformed into a vector. Then, we measure the similarity by the cosine between the encoded vectors.

In Fig. E.1c, we provide similarity scores between target objects and background pixels on HSI data. Since there has been no general framework measuring the similarity between reflectance patterns, we measure the similarity by the average cosine between pixel-wise reflectance. Although the similarity metrics used in RGB and HSI data are different, the relative score in the same metric shows the similarity in different categories.

As shown in the figure, we confirm that the background pixels with and without sea surface effects have similar reflectance patterns, but they are dissimilar in RGB data. Furthermore, the floating matters and background pixels with sea surface effects have similar RGB data, whereas their reflectance patterns are distinguishable.



(a) Randomly sampled RGB data.



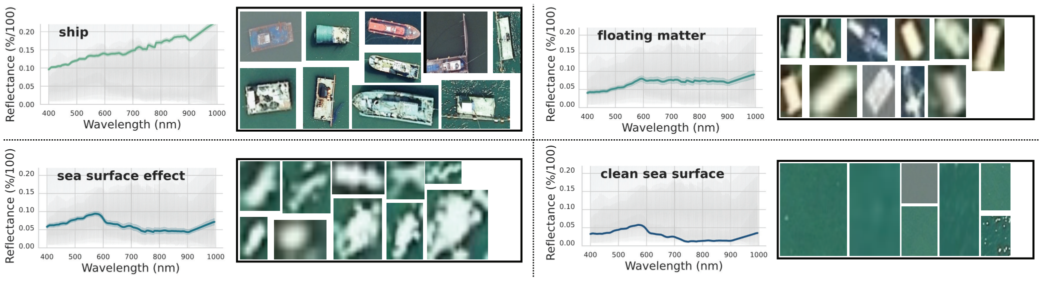
(b) Randomly sampled HSI data.

Figure D.1: Examples of annotated training data. (a): Randomly sampled annotated RGB data are depicted. (b): The HSI data corresponding to the RGB data in (a) are visualized. For the visualization, we represent the HSI data into gray-scale images, in which the pixel values are the ratio of 25-th and 72-nd channels.

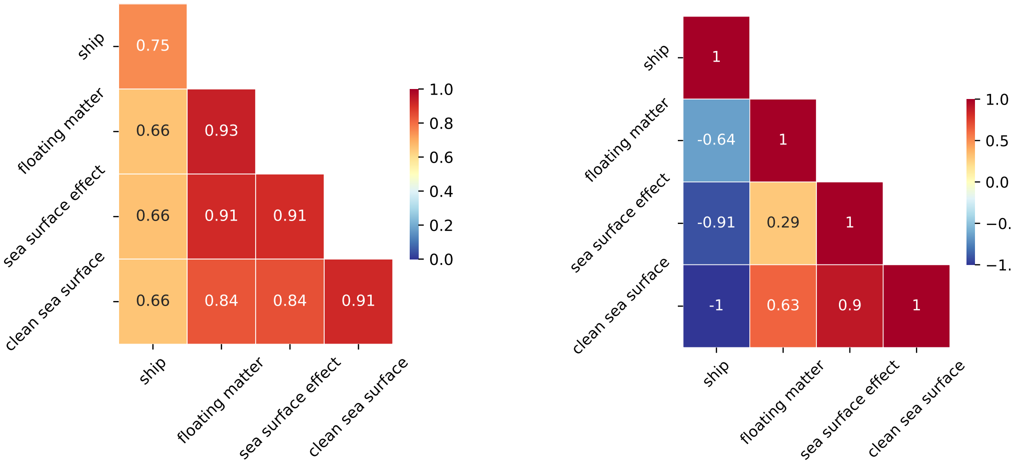
# F Implementation Details

In this section, we introduce details of the experiment settings that are not mentioned in the main paper.

**Early fusion and late fusion** Figure F.1 illustrates the proposed DoubleFPN, the canonical FPN, and early/late fusion methods. In (a), the canonical FPN utilizes either RGB or HSI data. In (b) and (c), canonical FPN methods with early/late fusion are depicted. In the early fusion method, the RGB and HSI data are resized and concatenated before the backbone layer. After the fusion, the feature maps are extracted in the same way as the canonical FPN. In the late fusion method, two FPN layers obtain feature maps independently, then the region proposal network (RPN) forwards object detection results. In (d), the DoubleFPN architecture is depicted, in which each level of FPN features represents the feature map of different scales. Then, an integrated feature map can be obtained by combining RGB feature maps and HSI attention maps taken from the same location.



(a) Reflectance patterns.

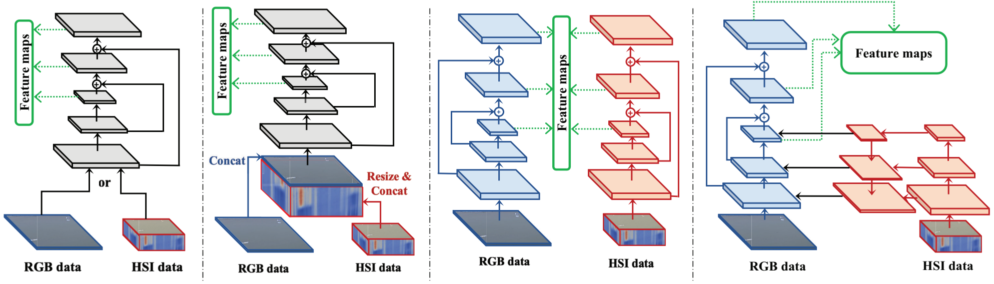


(b) Similarity in RGB data.

(c) Similarity in HSI data.

Figure E.1: Category-wise similarity analysis of the M$^2$SODAI dataset. (a): We depict the reflectance patterns of ships, floating matter, clean sea surface, and sea surface effects (wave, sunlight, oil, and etc.). (b): The category-wise similarity in RGB data is depicted, in which the RGB data are transformed into encoded vectors. (c): The category-wise similarity in HSI data is depicted, in which the similarity is measured by the cosine between the reflectance patterns.

**Head layer** Here, we introduce an application of our method to Faster R-CNN [40]. The DoubleFPN architecture forwards output feature maps $P_1, ..., P_N$ to the RPN. First, the RPN proposes anchor boxes for output feature maps $P_1, ..., P_N$. In the RPN, each feature map $P_i$ is fed into $3 \times 3$ convolution layer and two posterity $1 \times 1$ convolution layers. Then, for the $i$-th feature map $P_i$, the RPN proposes anchor boxes having areas of $32^2 \cdot i^2$ with three ratios {1:1, 2:1, 1:2}. For instance, there are three anchor boxes having areas of $128^2$ with $k$ different ratios ({1:1, 2:1, 1:2}, $k = 3$) over the feature map $P_3$. Then, bounding box regression values and scores of each anchor box are obtained in the *cls* and *reg* layers. During the training, the anchor boxes having intersection-over-union (IoU) ratios above 0.7 with a ground truth bounding box are considered *positive* anchor boxes. Also, the other anchor boxes having IoU ratios under 0.3 are considered negative anchor boxes. We note that the anchor boxes, neither positive nor negative, do not join in the training procedure. Let the score and bounding box of the $i$-th anchor box be $p_i \in [0, 1]$ and $t_i \in \mathbb{R}^4$. For training the RPN, we employ the multi-task loss function in [40], which trains the model parameters to reduce the score of the negative anchor boxes and let the positive anchor boxes be the ground-truth anchor boxes. Second, in the Faster R-CNN [40], the proposed anchor boxes are projected into feature maps $P_i$, *i.e.*, RoI pooling. We note that an RoI box of width $w$ and height $h$ is assigned to $k$-th feature map $P_k$, where $k = \max\{0, \lfloor \log_2(\sqrt{wh}/28) \rfloor\}$. For classification, the RoI pooling layer employs max pooling to convert the positive anchor boxes to fixed-size features ($7 \times 7$). After the RoI pooling, the output is flattened and forwarded to a series of fully-connected layers. At the end of the fully-connected layers, we have the class logits and bounding box predictions. If our task is to detect and classify $c$-class objects, class logits and bounding box predictions for $i$-th anchor box are defined by $q_i \in \mathbb{R}^{c+1}$



(a) Canonical FPN (b) Canonical FPN + Early Fusion (c) Canonical FPN + Late Fusion (d) DoubleFPN (Middle Fusion)

Figure F.1: Illustration of the multi-modal object detection methods. (a): The canonical FPN [26] obtains feature maps of RGB or HSI data for diverse levels of spatial scale. (b): early fusion [11] method with a convolution layer. (c): The late fusion methods [43, 44] combine the RGB and HSI data in the backbone/head layers, respectively. (d): The DoubleFPN fuses feature maps of the RGB and HSI data in the neck layer (FPN).

and $t'_i \in \mathbb{R}^{c\times 4}$ if the box is a positive anchor. To train the RoI head, we employ the multi-task loss function in [40].

**Pre-trained model weights** In our experiments, the ResNet models [19] are used as our backbone layers. Most existing works initialize the backbone layers with pre-trained weights to further improve the detection accuracy. For the RGB input, there have been abundant datasets for pre-training the backbone layer; hence, it is easy to find a proper pre-trained model. However, there has yet to be a pre-trained model for HSI input. Thus, for fairness of the training, we trained the object detection methods from scratch.[6]

**Hyperparameter settings** For the FPN layer, the output channel is 256, *i.e.*, $d = 256$, where the total number of output feature maps is $N = 5$. In the RPN layer, the anchor generator creates anchors with three ratios {1:1, 2:1, 1:2} and five scales {32, 64, 128, 256, 512}. For other settings, we follow the hyperparameters of the canonical FPN framework [26], where the scheduler and normalizer settings follow the training from scratch settings in [20]. Keep the learning rate at 0.02 for the first 64 epochs, and reduce the learning rate by a factor of 1/10 each in the 65th and 71st epochs.

**Incremental PCA** In order to reduce the dimensionality of the HSI data, we leverage principal component analysis (PCA) into the training dataset. Since the data size is too large, we implement the incremental PCA with a batch size of 50. Before applying the PCA, we normalize the reflectance $r_{x,y,c}$ in the $(x,y)$-th pixel of the $c$-th channel by

$$\hat{r}_{x,y,c} = \frac{r_{x,y,c} - \mu_c}{\sigma_c}, \tag{1}$$

where $\mu_c$ and $\sigma_c$ denote the mean and standard deviation of the reflectance of the $c$-th channel.

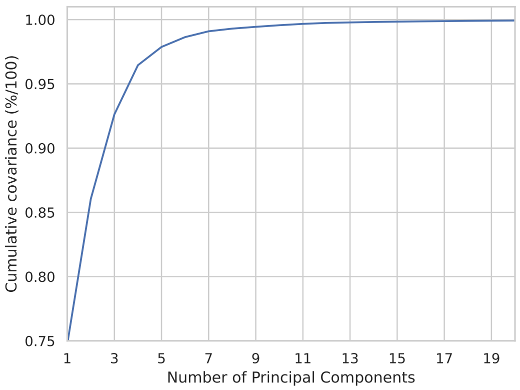


Figure F.2: Cumulative variance of the principal components. The principal components are obtained via incremental PCA.

In Fig. F.2, we depict the relative cumulative variance of the principal components. As increasing the number of principal components, the cumulative variance increases. As we can see, the cumulative variance with 20 principal components achieves about 99.99% of the total cumulative variance. That is, 99.99% of the information in the HSI data can be obtained by extracting

---

[6] With carefully pre-trained models, the AP performance can be further enhanced; however, we compare the performance without a pre-trained model.

20 principal components out of 127 channels. Hence, in our experiments, we reduce the dimensionality of the HSI data by leveraging sufficiently many principal components (30 components) via the incremental PCA method. The incremental PCA method is built based on the scikit-learn library [36].

**Preprocessing** For data augmentation, we utilize the following four functions:

- **Normalization (RGB/HSI):** The RGB and HSI data are normalized by the mean and standard deviation of each channel.
- **Incremental PCA (HSI):** The dimensionality is reduced by the incremental PCA method.
- **RandomFlip (HSI/RGB):** The RGB and HSI data are flipped vertically/horizontally with prob. 50%.
- **Resize (HSI/RGB):** The RGB and HSI data are resized by 1600×1600 pixels and 224×224 pixels, respectively.

We implemented experiments with other augmentation functions, such as random crop with minimum IoU threshold and padding; however, there are no notable differences in experiment results.

The Faster R-CNN detector is built based on the open-sourced MMDetection library [6], in which several functions are added and modified for our multi-modal object detection frameworks. Also, our code will be publicly available.

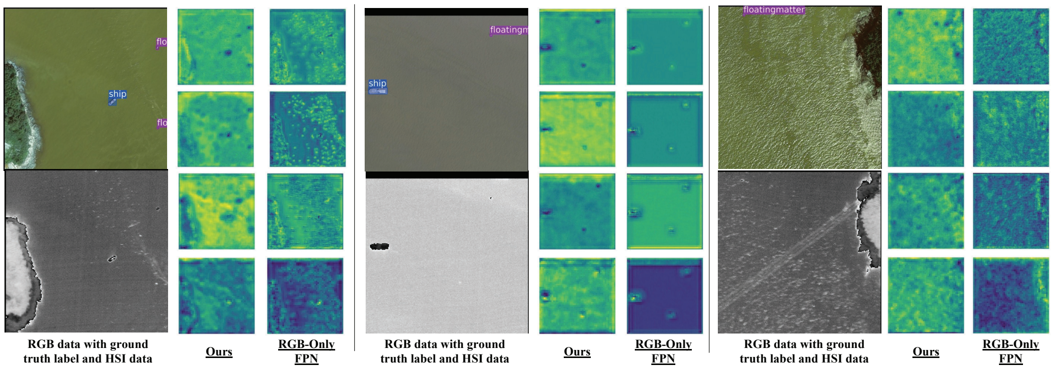
## G Extra Experimental Results

Table G.1: Benchmark result on the M$^2$SODAI dataset with the DoubleFPN and the uni-modal and multi-modal baseline methods. We compare the Faster R-CNN-based object detection methods for various backbone layers.

| **neck layer** | **backbone** | mAP | AP$_{@.5}$ | AP$_{@.75}$ | Ship | Float. Mat. | AP$_s$ | AP$_m$ | AP$_l$ |
|---|---|---|---|---|---|---|---|---|---|
| **DoubleFPN (ours)** | ResNet-34 | 42.0 | 82.6 | **38.3** | 53.7 | **33.2** | **33.3** | 38.7 | 58.3 |
| | ResNet-50 | **44.4** | **84.8** | 39.3 | 55.7 | **33.1** | **35.2** | 41.7 | **61.4** |
| | ResNet-101 | 43.7 | **86.1** | 37.3 | **55.5** | 31.4 | **26.3** | 42.3 | 60.3 |
| | ResNeXt-50 | **43.5** | **85.7** | **40.2** | 53.4 | **33.5** | 23.9 | 42.9 | 58.6 |
| FPN (RGB) | ResNet-34 | 38.6 | 74.2 | 32.1 | 51.2 | 25.3 | 16.3 | 40.8 | 53.6 |
| | ResNet-50 | 38.8 | 77.0 | 33.3 | 52.4 | 25.2 | 18.3 | **44.8** | 55.6 |
| | ResNet-101 | 39.7 | 74.7 | 36.6 | 54.6 | 24.9 | 18.7 | **44.4** | 60.2 |
| | ResNeXt-50 | 38.9 | 75.4 | 37.0 | 54.2 | 23.5 | 13.0 | **43.5** | 60.1 |
| FPN (HSI) | ResNet-34 | 8.0 | 23.3 | 2.4 | 15.9 | 0.0 | - | - | - |
| | ResNet-50 | 7.8 | 23.2 | 2.9 | 15.8 | 0.0 | - | - | - |
| | ResNet-101 | 8.5 | 25.6 | 4.1 | 17.1 | 0.0 | - | - | - |
| | ResNeXt-50 | 8.0 | 24.5 | 3.2 | 16.1 | 0.0 | - | - | - |
| UA-CMDet | ResNet-34 | **42.1** | 82.5 | 37.3 | **55.0** | 29.2 | 22.2 | 37.7 | **60.4** |
| | ResNet-50 | 42.9 | 84.0 | **40.0** | **55.9** | 29.8 | 20.8 | 43.0 | 60.8 |
| | ResNet-101 | 43.6 | 84.3 | 39.9 | 54.5 | 33.7 | 26.0 | 42.1 | **62.6** |
| | ResNeXt-50 | 43.3 | 83.8 | 36.3 | **55.2** | 31.4 | **28.6** | 42.5 | **61.1** |
| DetFusion | ResNet-34 | 41.8 | **84.6** | 33.7 | 53.0 | 30.5 | 28.8 | 40.0 | 57.3 |
| | ResNet-50 | 42.0 | 84.3 | 35.4 | 53.5 | 30.5 | 24.2 | 41.9 | 61.1 |
| | ResNet-101 | **44.2** | 84.3 | **42.1** | 55.0 | 33.5 | 19.3 | 42.0 | 60.9 |
| | ResNeXt-50 | 42.5 | 82.8 | 37.4 | 54.3 | 30.7 | 18.4 | 44.4 | 59.8 |
| Early fusion | ResNet-34 | 41.8 | 83.4 | 35.9 | 52.8 | 30.8 | 22.5 | **43.5** | 57.0 |
| | ResNet-50 | 42.9 | 83.0 | 37.6 | 54.2 | 31.5 | 18.9 | 44.1 | 59.7 |
| | ResNet-101 | 44.1 | 85.5 | 38.2 | 54.1 | **34.2** | 21.7 | 43.4 | 60.8 |
| | ResNeXt-50 | 42.2 | 84.0 | 36.1 | 53.4 | 30.9 | 21.0 | 42.1 | 58.7 |

*Best: **bold and underline**, second-best: underline (for each backbone layer).

In this section, we furnish the following additional results: i) performance benchmark with different backbone layers, ii) object detection examples in maritime rescuing applications, iii) feature map analysis, and iv) additional samples of the object detection results.



RGB data with ground truth label and HSI data | Ours | RGB-Only FPN | RGB data with ground truth label and HSI data | Ours | RGB-Only FPN | RGB data with ground truth label and HSI data | Ours | RGB-Only FPN

Figure G.2: Feature map analysis. The strongest feature among channels is selected for each pixel for visualization. In the left and right figures, the feature maps of our method and the RGB-only uni-modal FPN are depicted, in which the input data are with sea surface effects. In the middle figures, the feature maps without sea surface effects are depicted. For the data in the clean sea (middle), both methods provide clean feature maps; however, with the sea surface effects (left/right), our method furnishes clearer feature maps than the RGB-only uni-modal FPN.

**Different backbone layers** In the main paper, we analyze the performance benchmark with the ResNet-50 backbone layer. Here, we evaluate the object detection methods using a lighter backbone model (ResNet-34), a heavier backbone model (ResNet-101), and another type of backbone model (ResNeXt-50).[51] Table G.1 establishes the benchmark results of our method, RGB/HSI-only uni-modal FPN, early fusion, DetFusion [44], and UA-CMDet [43]. Similar to the main paper, we analyze the performance benchmark with average precision (AP)-based metrics. As the backbone layers become heavier, most AP-based benchmark results are generally enhanced.

Similar to the results in the main paper, our method generally outperforms the baseline methods in all AP metrics except AP$_m$. The RGB-only uni-modal FPN has a comparable AP in large objects (or ships); however, this RGB-only method suffers from finding smaller objects.

**Maritime rescuing applications** During the data acquisition, we collected RGB and HSI data containing clothed mannequins, lifeboats, and life tubes on the sea for maritime rescuing applications. Figure G.1 shows an example of those target objects on our dataset. In the figure, there are five floating matters and one ship. On the right side, the cropped images of the ground truth annotation and our detection result are depicted. As a result, our method finds all the target objects in maritime rescuing applications.

**Feature map analysis** Here, we analyze additional experimental results for feature maps of the DoubleFPN method. Figure G.2 shows the feature maps of the DoubleFPN method and the RGB-only uni-modal FPN. The figures in the $n$th row represent the $n$th feature maps; in other words, the figures in the $n$th row represent the feature maps on the scale of $(1/2)^{n+1}$. As shown in the figure, when there are sea surface effects, our method can extract the feature map for the target object more clearly than the existing RGB-only FPN method.

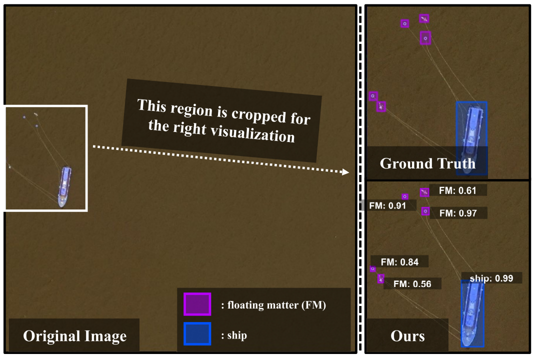


Figure G.1: An application to maritime rescuing systems. The floating matters and ship are labeled by **purple** and **blue** colors, respectively. For comparison, the ground truth bounding boxes and our detection results are depicted.

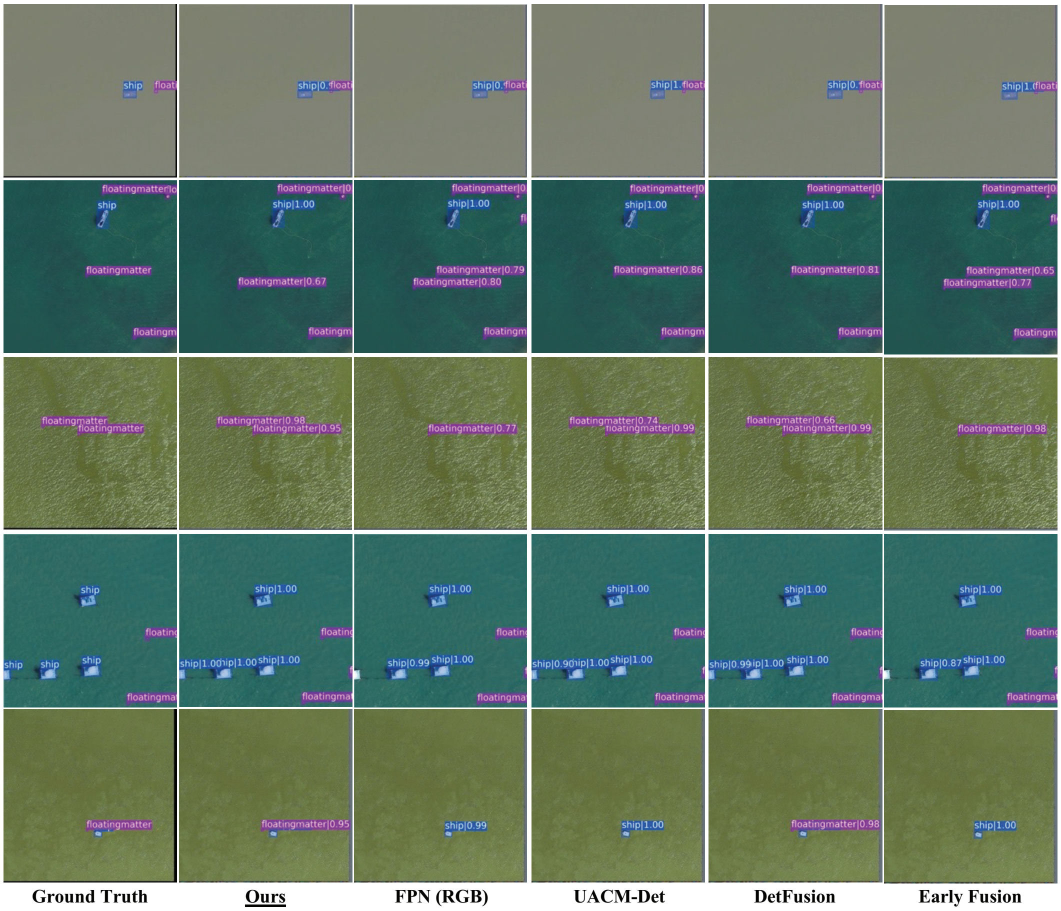


Figure G.3: Detection results on data with sea surface effects. The figures in the first column depict the ground truth bounding boxes. The other figures show the estimated bounding boxes of the object detection methods.

**Samples of the detection results** For more plentiful graphic results, we provide some samples of the object detection results of our method and other baseline methods. In Fig. G.3, object detection results of the DoubleFPN, RGB-only FPN, DetFusion, UA-CMDet, and early fusion are depicted. The figures in each row describe the detection results on the same input data.

**Complexity Analysis** Table G.2 shows the computational complexity of the object detection methods in order of floating point operations (FLOPS). All the fusion methods' computational complexity is comparable, even though the DoubleFPN has dominating APs. Owing to the nature of the fusion process, which entails both a more prolonged inference time and increased GPU usage, we have identified the complex reduction of this process as a challenge for future consideration.

Table G.2: Complexity benchmark in order of floating point operations (FLOPS), number of parameters, inference time, and GPU usages.

| neck layer | Complexity (GFLOPS) | Number of Parameters ($\times 10^6$) | Inference Time (msec) | GPU Usage (GB) |
|---|---|---|---|---|
| **DoubleFPN (ours)** | 1230.75 | 68.29 | 123.41 | 19.2 |
| FPN (RGB) | 968.64 | 41.35 | 95.22 | 16.4 |
| UA-CMDet | 1288.87 | 68.89 | 129.81 | 19.8 |
| DetFusion | 1386.75 | 68.48 | 135.15 | 21.0 |
| Early fusion | 1230.95 | 68.29 | 135.45 | 19.5 |