Degree Project in Computer Science and Engineering, specializing in Machine Learning

Second cycle 30 credits

Multi-Resolution Inference of Bathymetry From Sidescan Sonar

ZHENGJIE JI

Stockholm, Sverige 2023

Multi-Resolution Inference of Bathymetry From Sidescan Sonar

ZHENGJIE JI

Master in Machine Learning
Date: December 21, 2022
Supervisor: Yiping Xie
Examiner: John Folkesson
School of Electrical Engineering and Computer Science
Swedish title: Flerupplöst slutledning av batymetri från Sidescan ekolod

Abstract

How to obtain complete and high-resolution bathymetry is an important research topic in the underwater domain. However, existing methods have certain shortcomings. Multibeam echosounder (MBES) can produce narrow beam range readings of the seafloor, but there is an interval between every two beams (between 10cm to 10m), and the resolution is low. Sidescan sonar can measure the seafloor in much higher resolution (down to below 1cm), but it is difficult to convert the sidescan into the bathymetry. Although several methods allow us to use physical models to estimate bathymetry from the sidescan, these methods are computationally difficult due to the growing amount of data. To bridge the gap, we propose a neural network-based system that can efficiently and accurately reconstruct high-resolution bathymetry from low-resolution bathymetry and the sidescan. In particular, the multi-resolution inference system can (1) efficiently extract the features of the sidescan sonar map; (2) reconstruct the high-resolution bathymetry using the extracted features and the input low-resolution bathymetry. Evaluations demonstrate that the inference system can reconstruct high-resolution bathymetry under different input settings.

Sammanfattning

Hur man får fram fullständig och högupplöst batymetri är ett viktigt forskningsämne inom undervattensdomänen. Befintliga metoder har dock vissa brister. Multibeam ekolod (MBES) kan ge avläsningar av havsbotten med smalt strålområde, men det finns ett intervall mellan varannan strålar (mellan 10 cm till 10 m), och upplösningen är låg. Sidescan ekolod kan mäta havsbotten i mycket högre upplösning (ned till under 1 cm), men det är svårt att omvandla sidoscanningen till batymetri. Även om flera metoder tillåter oss att använda fysiska modeller för att uppskatta batymetri från sidoskanningen, är dessa metoder beräkningssvåra på grund av den växande mängden data. För att överbrygga klyftan föreslår vi ett neuralt nätverksbaserat system som effektivt och exakt kan rekonstruera högupplöst batymetri från lågupplöst batymetri och sidoskanningen. I synnerhet kan flerupplösnings-inferenssystemet (1) effektivt extrahera egenskaperna hos sidoavsöknings-ekolodskartan; (2) rekonstruera den högupplösta batymetrin med hjälp av de extraherade funktionerna och den ingående lågupplösta batymetrin. Utvärderingar visar att inferenssystemet kan rekonstruera högupplöst batymetri under olika ingångsinställningar.

Contents

Chapter 1

Introduction

Bathymetry is to measure the underwater depth of ocean floors. By measuring the seabed topography, many forms of underwater research and applications can be carried out. Among them, the use of multibeam echo sounder systems (MBES) and sidescan sonar systems is commonly used.

Multibeam echo sounder systems use a set of transmitters to transmit a wide coverage of sound waves to the ocean floor and use a set of receivers to receive the reflected waves. The transmitted sound waves and the reflected sound waves will intersect vertically, and the depth values of the seabed in the perpendicular plane can be measured. The system can accurately detect underwater objects along the route by periodically transmitting sound waves. We are then able to draw a 3D feature map of the seabed topography by processing the collected data. The advantage of the multibeam echo sounder systems is that the bathymetry on the vertical plane can be measured with high precision. The disadvantage is that there is a large interval between the vertical planes, and the detailed information of the terrain is missing in these intervals.

Sidescan sonar systems use reflections of sound waves to obtain images of underwater topography. The system sends sound waves to the seabed from the transmitter to both sides. Due to the different shapes and materials of the underwater terrain, the reflected sound waves have different intensities. The sidescan sonar system will then process the intensity signals into 2D grayscale images with different shades. Combined with the underwater vehicle's position and speed information, we are able to obtain features describing the underwater terrain within the detection range. The advantage of the sidescan sonar system is that it can perform a wide range and high-resolution underwater terrain inspections. The disadvantage is that the intensity image generated by the sidescan is different from the bathymetry, and there is no direct conversion between the two images.

Although there are several methods that allow us to use physical models to estimate bathymetry from the sidescan, these methods are computationally difficult due to the growing amount of data. Therefore, we urgently need a method to obtain high-resolution bathymetry with existing equipment.

To leverage the rich information provided by the sidescan, we propose a neural network-based system that can efficiently and accurately reconstruct high-resolution bathymetry from low-resolution bathymetry and the sidescan, assuming that our autonomous underwater vehicle (AUV) can obtain both low-resolution bathymetry and the sidescan at some location.

1.1 Challenges

There are two main challenges in designing such a system: (1) We need to take advantage of the detailed information contained in the sidescan. Although the sidescan cannot be directly converted to bathymetry, the sidescan contains full details describing the terrain. With these details, we can recover high-resolution (equal to the resolution of the sidescan) bathymetry. Therefore, the system needs to be able to accurately extract the features from the sidescan, which is the key to solving the problem. We also need to correspond the bathymetry and sidescan according to the position coordinates, so that the complete bathymetry can be obtained through batch reconstruction. (2) We do not have ready-made datasets. To train the neural network model, we need data containing low-resolution bathymetry, high-resolution bathymetry, and high-resolution sidescan. The triples need to be able to correspond in position coordinates. We also need to record the parameters of the AUV when gathering data. A complete dataset is essential when testing and evaluating the system's performance.

1.2 Contribution

In this project, the main contribution is: We design a multi-resolution inference system that can extract features in the sidescan and rely on these features to reconstruct high-resolution bathymetry on low-resolution bathymetry. The design of the neural network structure is as follows: the neural network adopts a recursive structure to extract features in the sidescan and uses shortcuts between neighboring layers to speed up learning the residual between the prediction and the ground truth. In particular, we use interpolated low-resolution bathymetry to provide a positional reference for the reconstruction.

1.3 Evaluation

We performed comprehensive evaluations of the multi-resolution inference system on the multi-resolution dataset from multiple aspects. The evaluation results demonstrate that: (1) The system can reconstruct bathymetry well. The system achieves 38.5328 PSNR score on the test set, which is a 13% improvement over interpolation-based methods. (2) The entire system pipeline is efficient. The average reconstruction time is 1826 ms. The CPU usage of the system during reconstruction is up to 5243 mib. Based on the above experimental results, our system can reconstruct high-resolution bathymetry well and efficiently.

1.4 Social Benefits, Ethics and Sustainability

In this section, we report the problems remain to be solved in this field and motivate our work accordingly.

1.4.1 Social Benefits

From the perspective of social benefits, as underwater terrain detection plays a very important role in ocean engineering, it would definitely promote economic development. Seabed measuring is widely used and is closely related to energy extraction. Not only that, seabed measuring can promote the advancement of maritime navigation technology and has a significant influence on both military and business applications. Here we listed a few representative application scenarios. First, terrain detection helps us conduct marine scientific research. Second, underwater terrain detection can help the work of the underwater navigation system, helping the safe navigation of both surface and underwater vehicles.Third, a complete and detailed seabed topographic map can help us better plan the laying of underwater pipelines. Fourth, detecting seabed topography will help us develop seabed resources, such as oil and gas exploration and mineral mining. Therefore, it is necessary to understand the changes in seabed topography in detail and draw detailed bathymetry maps, which is one of the main motivations of our work.

1.4.2 Ethics

For ethical aspects, underwater terrain detection also plays an important role in environmental monitoring and protection. When exploiting seabed resources (such as natural gas, oil, and other energy resources), it is an essential prerequisite to accurately describe the topography of the seabed. Otherwise, we may not be able to produce accurate estimates of the distribution of seafloor resources, and we are likely to have serious consequences in the mining process. Improper resource exploitation may lead to environmental pollution (such as oil spills). More seriously, inappropriate resource exploitation may lead to serious production safety accidents. Good underwater terrain detection can help us exploit resources more safely and rationally, thereby protecting marine resources, preventing marine pollution, maintaining ecological balance, and protecting the safety of maritime workers.

1.4.3 Sustainability

From the angle of sustainability, our work aims at using limited resources to achieve satisfying performance in underwater terrain detection. At present, seabed measuring technology based on AUVs has emerged rapidly. However, detecting seafloor topography with a large number of sensors has a high cost. First, underwater detectors are expensive. Second, if high-precision data is required, we may need more sensors to collect data. This leads to very high energy consumption. Third, bathymetric data processing requires high manpower. All the while, we need to manually preprocess the raw data. This results in very slow fetching of data that can be used. For the above reasons, we propose a scheme in this project that uses a machine learning approach to reconstruct high-quality, high-precision bathymetry from data obtained from existing sensors.

Chapter 2

Background and Related Work

2.1 Underwater Topographical Survey

The underwater topographical survey is the work of measuring underwater landscape and features. The results of underwater topographic surveys are usually underwater topographic maps (in the form of bathymetry). Compared to the topographical study of land, it is more complicated because AUV (Autonomous Underwater Vehicle) needs to be navigated in water and thus needs to have strong stability. Among them, the sidescan sonar and the multibeam echosounder (MBES) are two widely used equipment mounted on AUV.

Sidescan Sonar. The sidescan sonar is a kind of underwater exploration equipment that uses sound wave reflection to provide a horizontal image of the seabed. It was first studied by German scientist Julius Hagemann who worked in the Navy Marine Defence Laboratory of the United States and developed the prototype of the sidescan sonar in 1958 [1]. The side scan sonar system is mainly composed of four parts: the positioning system, the rotating drum, the central computer, and the control unit. Sidescan sonar systems use reflections of sound waves to obtain images of underwater topography. The system sends sound waves to the seabed from the transmitter to both sides. Due to the different shapes and materials of the underwater terrain, the reflected sound waves have different intensities. The sidescan sonar system will then process the intensity signals into 2D grayscale images with different shades. Combined with the underwater vehicle's position and speed information, we are able to obtain features describing the underwater terrain within the detection range.

Multibeam Echosounder. The echosounder uses sound waves to measure the distance from the vehicle to the bottom. Multibeam echo sounder systems use a set of transmitters to transmit a wide coverage of sound waves to the ocean floor and a set of receivers to receive the reflected waves. The transmitted sound waves and the reflected sound waves will intersect vertically, and the depth values of the seabed in the plane perpendicular can be measured. The system can accurately and efficiently detect underwater objects along the route by periodically transmitting sound waves. We can draw a 3D feature map of the seafloor topography by processing the collected data. The advantage of the multibeam echo sounder systems is that the bathymetry on the vertical plane can be measured with high precision. The disadvantage is that there is a large interval between the vertical planes, and the detailed information of the terrain is missing in these intervals.

Prior to this project, we already tried to apply CNN-based method to infer depth contours from sidescan. In the paper [2], the author trained two pixel-to-pixel CNNs for regression and a conditional GAN for calculating the loss. The author concluded that it is possible to generate bathymetry using sidescan sonar images. However, the performance is not optimized due to the limited number of samples. Also, we observed sharp changes in the rocky areas, which can be further improved.

2.2 Feature Extraction and Feature Selection

Feature Extraction. Feature extraction has many applications in machine learning. Feature extraction refers to extracting informative features from a raw dataset (features are usually represented as tensors). Features can help us better describe and analyze data. For example, in the field of image processing, feature extraction algorithms are used to extract information such as shape, color, figure, etc., from the original image. The extracted features make it possible to accomplish various image-processing tasks, such as image classification, object recognition, object tracking, etc. Common feature extraction methods are: PCA [3], LDA [4], MDS [5], kernel methods [6], and neural network based methods. Detailed neural network based feature extraction algorithms will be discussed in Section 2.3.

Feature Selection. Feature selection is one of the most important methods in feature engineering. Feature selection is first proposed to solve the problem of overfitting. An overfitted model will perform well on the training set, but the performance will drop significantly on the test set. There are many different reasons for model overfitting, such as poor model generalization ability, improper dataset partitioning, etc. By reducing the number of irrelevant and redundant features, feature selection can effectively improve model accuracy and reduce the time required for training. Not only that, but feature selection can also help us better interpret the data and build more relevant models to the dataset. Existing feature selection methods include (1) filter method; (2) wrapper method; and (3) embedded method [7].

2.3 Artificial Neural Network

The artificial neural network is the core of deep learning algorithms. It simulates the way biological neurons transmit signals to each other. An artificial neural network is an adaptive system that can be used to estimate or approximate a mathematical model.

In particular, when an artificial neural network is used to extract features (of an image), we call it a backbone network. A backbone is used to extract high-dimensional features from images, thereby helping the model to complete various downstream tasks, such as image classification, object recognition, object tracking, etc. Popular backbones include VGGNet [8], GoogLeNet [9], ResNet [10], DenseNet [11], EfficientNet [12], ViT [13], and etc.

Deep Residual Learning. The artificial neural network contains many artificial neurons, which jointly perform calculations. The depth of the neural network is a great influencing factor in the model's performance. As the number of layers in the network increases, the network can extract more complex features from the data, leading to better results for the model.

However, the degradation problem of deep neural networks makes them not easy to train. In reality, when the network depth increases to a certain extent, the model's performance will actually decrease. The reasons for the degradation problem include (1) Overfitting caused by large parameter amount. The more layers of the model and the larger the number of parameters, the weaker the generalization ability of the model; (2) the problem of gradient disappearance and gradient explosion caused by large layer number; (3) the loss of information caused by nonlinear changes in the deep neural network.

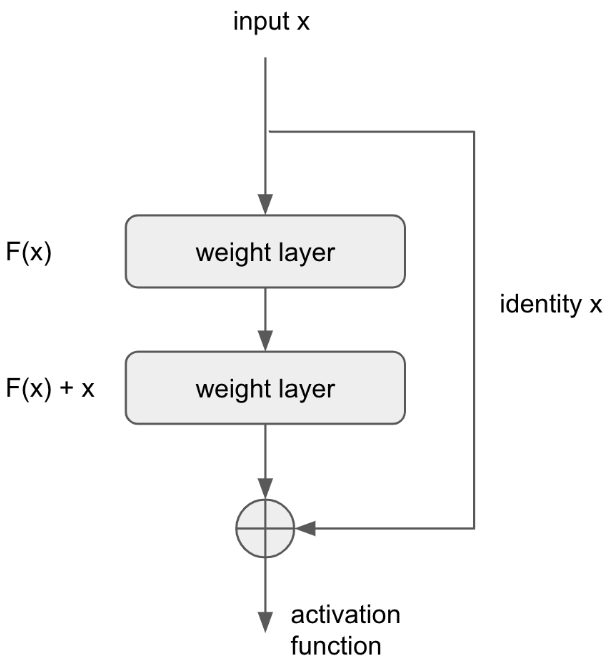


Figure 2.1: A simple residual block [10]

Although some technical means such as batch normalization [14] can alleviate the degradation problem of deep neural networks, the degradation problem is still severe when the number of deep network layers is large. Deep residual learning (ResNet) [10] was first proposed by Kaiming He in 2015, aiming to solve the degradation problem. The ResNet refers to the structure of the VGG [8] network and modifies it based on it. The core component of the ResNet is the residual unit, which utilizes a short-circuit mechanism to make it easier to learn features, thereby providing better performance.

2.4 Super-Resolution

Super-resolution (SR) reconstruction is one of the most popular research areas in computer vision. Super-resolution reconstruction techniques are used to improve the resolution of the original image using either hardware or software techniques.

Super-resolution reconstruction techniques are usually used to deal with the following two situations (1) Single-frame image reconstruction from single-frame images. Input a two-dimensional image, and the system will output a two-dimensional image of higher resolution. (2) Reconstruct a super-resolution single-frame image from multiple sequential images. Input a two-dimensional image and its adjacent images in continuous time, and the system will fill the high-resolution image using the complementary information from all images.

Our application case is different from the above two. We input a single frame of a two-dimensional image (i.e., the bathymetry) and rely on additional reference information corresponding to this image (i.e., the sidescan sonar) to reconstruct a higher resolution image.

Frequency Analysis-Based Super-Resolution. Tsai and Huang [15] first proposed SR in 1984, using a re-construction method based on frequency analysis. The observation model can be represented as matrices. By under-sampling the sequential images, it is possible to use a matrix to represent the frequency-domain phase change relationship corresponding to the time-domain shift between frames. After solving for the coefficients of the equations, we can obtain the Fourier transform of the target high-resolution image.

Spatial Analysis-Based Super-Resolution. Some other methods are based on spatial analysis, such as non-uniform interpolation based methods [16]. The non-uniform interpolation method is the most simple and intuitive high-resolution image reconstruction method. Generally speaking, image reconstruction using non-uniform interpolation includes three steps: motion estimation, non-uniform interpolation, and deblurring. First, we estimate relative motion information between images. Then, we interpolate the image to get a high-resolution image. Finally, we perform post-processing on high-resolution images, such as deblurring and noise reduction, to further improve the quality of the images.

Learning-Based Super-Resolution. Learning-based methods are increasingly common in computer vision, as well as in super-resolution research. The learning-based method [17] mainly make use of the similarity in high-frequency details of different images. Learning-based algorithms extracts the relationship between high-resolution and low-resolution images for guiding the the reconstruction of high-resolution images.

Deep Learning-Based Super-Resolution. Different from the previous method, deep learning-based super-resolution focus on the model design and the design of loss function to optimize the performance. Among the deep learning methods, SRCNN [18] is the most representative one. The model first pre-process the low-resolution images and then feed the processed images into a neural network to extract high-frequency information representing image details. After extracting the features of the image, we use a convolutional neural network to reconstruct the high-resolution image. With a well-designed loss function, we can use the back-propagation algorithm to update the network parameters to train the model.

Chapter 3

Methods

In this section, we present the design details of the multi-resolution inference system, including the dataset construction and the model design.

3.1 System Overview

Figure 3.2 shows the architecture of the system. The Multi-Resolution Inference System takes the sidescan and low-resolution bathymetry as input. Then, the system generates and outputs high-resolution bathymetry. First, we use residual blocks to extract features from the sidescan and low-resolution bathymetry. The figure shows only one residual block for each of the two feature extractors. In actual implementation, five residual blocks are connected sequentially to increase the depth of the network and thus improve the performance. The system then concatenates the extracted features and feeds them into another convolutional network to generate high-resolution bathymetry. We add a shortcut from the low-resolution bathymetry to the output. This allows the neural network to learn the residual of the bathymetry, thereby improving convergence speed.

3.2 Demonstration Example

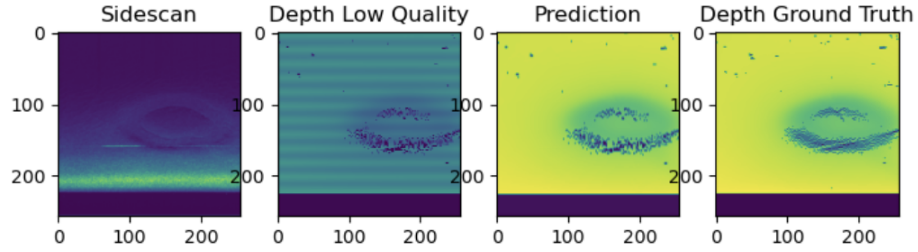


Figure 3.1: Demonstration of the input and output images

Figure 3.1 shows an example of image reconstruction. The leftmost image is the echo intensity recorded by the sidescan sonar system. Dark pixels represent high echo intensity. The resolution of the image is 256×256. The second image is a low-resolution bathymetry. There are 256 depth values on each row. Missing data between the two beams cause a certain distance between the rows. In the example, there is a 4-pixel-wide gap between every two rows. The third image is the bathymetry prediction output by the multi-resolution inference system, which outperformed the interpolation-based method. The fourth image is the ground truth bathymetry with a resolution of 256×256. We conclude that the system can reconstruct high-resolution bathymetry accurately.

3.3 Design of the Inference System

In this section, we present the design details of the multi-resolution inference neural network model, including the underlying feature extractor and the different techniques involved. The structure of this model is shown in the Figure 3.2.

The input to the neural network is a grayscale low-resolution bathymetry of size 256×256 and the sidescan. We perform column-wise linear interpolation on the input low-resolution bathymetry in advance. As the input is similar to the output, interpolation makes it easier for model training.

We perform feature extraction on the low-resolution bathymetry and the sidescan. We use multiple layers of residual blocks to do this. Each residual block contains two convolutional layers and uses a shortcut between the input and output. Each convolutional layer receives 128 channels of input and returns 128 channels of output. The input and output are of the same sizes as the original image. Five residual blocks with the same structure are connected sequentially, increasing the network's depth and allowing the network to extract features from the input more accurately.

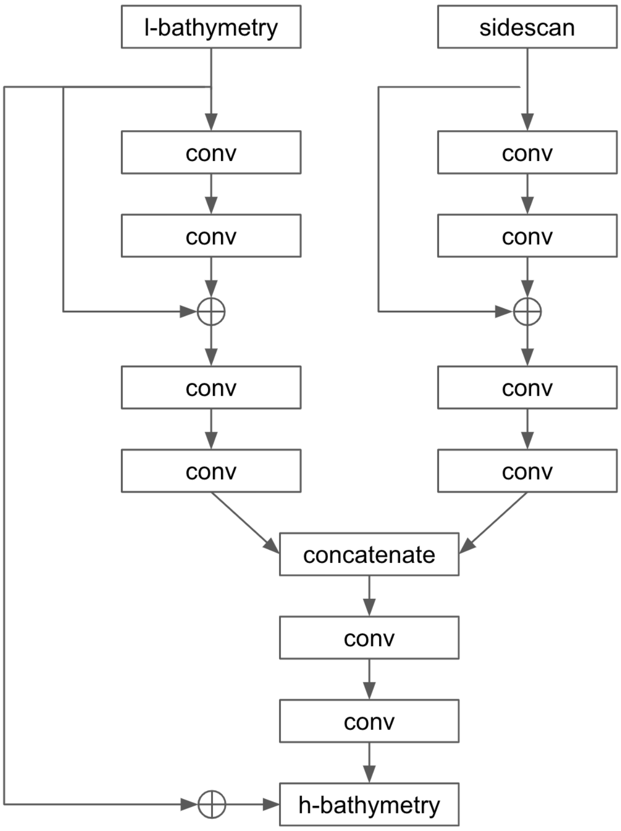


Figure 3.2: Architecture of the multi-resolution inference neural network model

After obtaining the feature maps of the low-resolution bathymetry and the sidescan respectively, we concatenate the feature maps and pass in another block to generate high-resolution bathymetry. In this block, the number of input and output channels increases to 256 from 128.

Additionally, we used a shortcut between the input's low-resolution bathymetry and the output's high-resolution bathymetry. This makes the neural network not directly learn high-resolution bathymetry but learn the residual between low-resolution bathymetry and high-resolution bathymetry. Due to the high similarity between input and output, learning the residual improves the convergence speed of model training. Alternative designs and corresponding evaluations are discussed in Section A.

Chapter 4

Experiments

4.1 Research Questions

We built the multi-resolution inference system based on the following tools. We use Python for the entire system, auvlib [19] for processing the multibeam data and the sidescan data, and PyTorch [20] for the neural network construction. Our goal is to answer the following research questions:

- RQ1: What is the performance of the multi-resolution inference system in reconstructing high-resolution bathymetry?
- RQ2: How will the input bathymetry resolution influence the reconstruction?
- RQ3: How efficient is the multi-resolution inference system?

RQ1 aims to evaluate the performance of the multi-resolution inference system in reconstructing high-resolution bathymetry by comparing the prediction of the system to the prediction generated by interpolation-based methods. RQ2 aims to evaluate the system's performance among different input bathymetry resolutions. RQ3 aims to evaluate the multi-resolution inference system's run time and memory usage.

4.2 Evaluation Setup

The experiment was carried out on a personal laptop which has an Intel(R) Core(TM) i7-8750H CPU, and an Nvidia RTX2060 GPU with 16GB RAM running Ubuntu 18.04.

We use the multi-resolution dataset generated by slicing the draping results mentioned in Section 3 as the test dataset. The dataset covered route 22 to route 182 in Area 1, including surveys from 20m depth to 70m depth.

In the evaluation part, we comprehensively studied the performance of the multi-resolution inference system. There are a total of 964 triplets in the multi-resolution dataset. In the experiments of RQ1 and RQ2, we use peak signal-to-noise ratio (PSNR) [21], structural similarity (SSIM) [22] and mean square error (MSE) as the evaluation metric. In the experiments of RQ3, we use time (ms) and memory usage (MiB) as the evaluation metric. Metrics for RQ1 and RQ2 are defined as follows.

MSE. Given a clean image I of size $m \times n$ and a noise image K, the MSE is defined as

$$\text{MSE} = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2.$$

PSNR. Based on the definition above, PSNR (dB) is defined as

$$\text{PSNR} = 10 \cdot log_{10}\frac{\text{MAX}_i^2}{\text{MSE}},$$

where MAX_i^2 is the maximum possible pixel values of an image, which is 1 in our case.

SSIM. SSIM is based on three measures between the two image samples x and y: luminance $l(x,y)$, contrast $c(x,y)$, and structure $s(x,y)$. The three measures are defined as follows:

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1},$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2},$$

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3},$$

where $c_3 = c_2/2$, μ_x is the mean value of x (same for y), σ_x^2 is the variance of x (same for y), σ_{xy} is the covariance of x and y. SSIM is defined as

$$\text{SSIM}(x, y) = l(x, y)^{\alpha} \cdot c(x, y)^{\beta} \cdot s(x, y)^{\gamma},$$

where α, β and γ are weights which are set to 1 as default.

The parameter settings for model training are shown in the table 4.1. We train all neural network models until convergence. Typically, ten epochs are enough for the learning curve to reach convergence. We use mean square error (MSE) as the loss function and stochastic gradient descent (SGD) as the optimizer.

Parameters	**Value**
Learning Rate	1×10^{-4}
Batch Size	4
Max Number of Epochs	20

Table 4.1: Parameter settings for training

It should be noted that we ignore pixels that have zero value on the ground truth image in training and evaluation. These pixels do not contain useful information and may lead to a decrease in the model's accuracy. We ignore those pixels by iterating over all pixels when calculating loss.

4.3 Multi-Resolution Dataset Construction

Our goal is to create a dataset for training the neural network part of the multi-resolution inference system. We first parse the raw data and create the basic dataset. Each data sample contains a high-resolution bathymetry and a high-resolution sidescan. Notice that each data sample should ensure that the location data can correspond. Working with the basic dataset, we can create different low-resolution bathymetry. Then, after filtering out low-quality images and data augmentation (flipping and adding Gaussian noise), we get the complete dataset. At last, we randomly shuffle the images and split the dataset. The training/validation/test split ratio is 8:1:1.

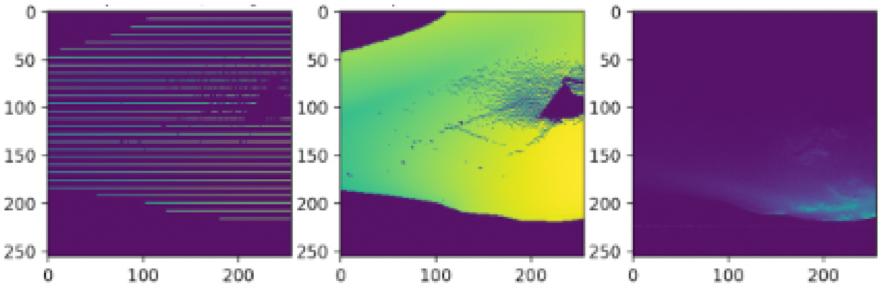
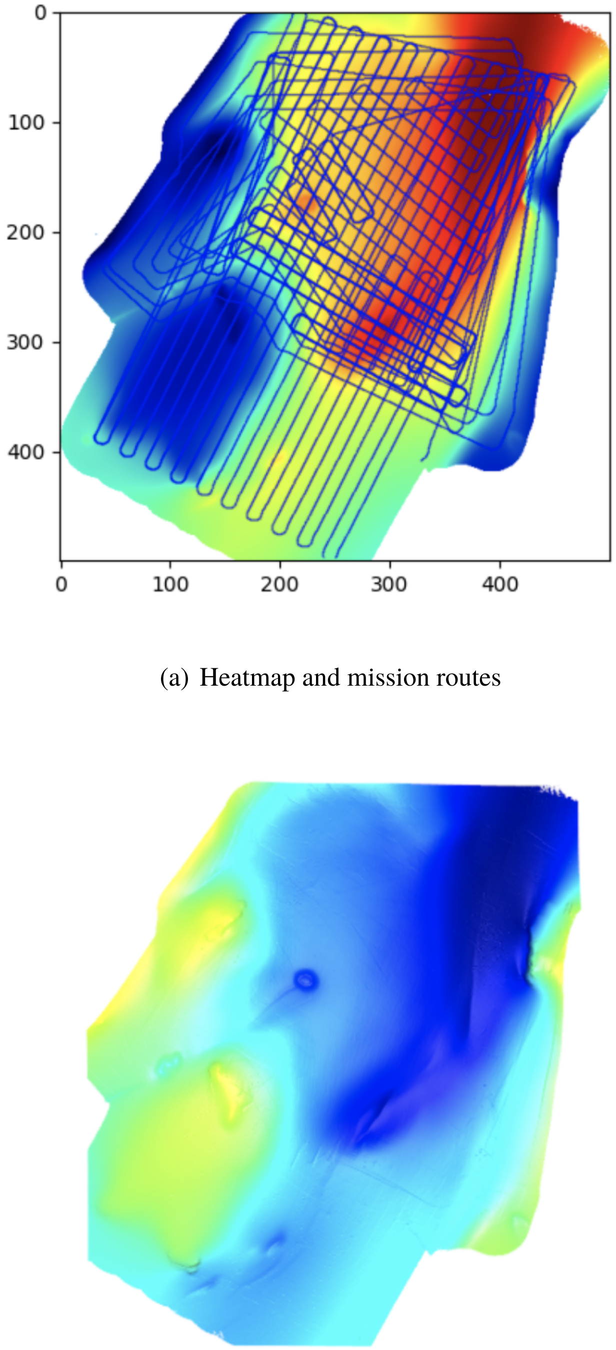


Figure 4.1: A sample in the dataset (from left to right: low-resolution bathymetry, high-resolution bathymetry and the sidescan)



(a) Heatmap and mission routes

(b) Mesh (height map)

Figure 4.2: Heat map and the big mesh (height map) generated from the parsed AUV data

Data Parsing. SMaRC provided us with the raw data recorded by AUV. There are four different types of data: positioning data, sound speed data, multibeam data, and sidescan data. Each mission has a unique mission number, and the data are divided into many files according to the mission number. To construct the basic dataset, we first load and parse all multibeam data. A large bathymetry mesh is generated by associating the multibeam data with the positioning data.

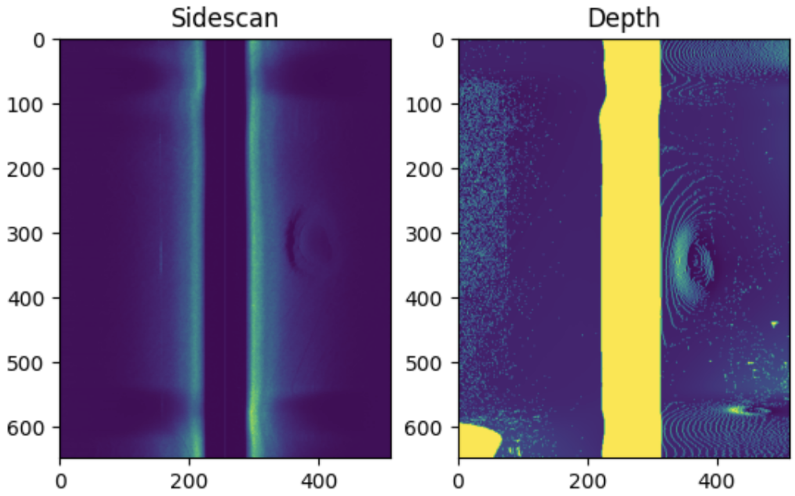
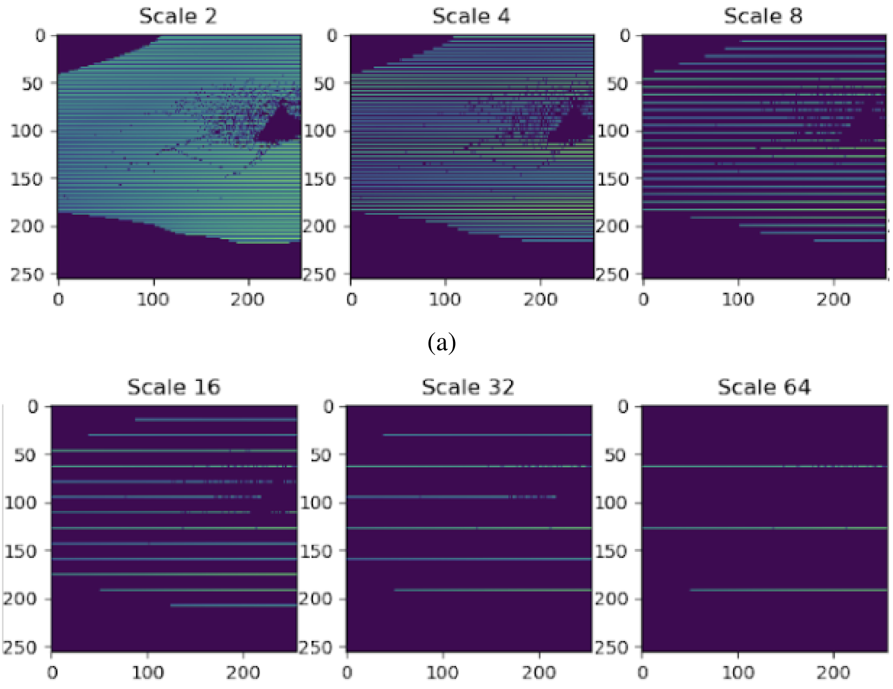


Figure 4.3: Mapped images (the sidescan and the bathymetry) after draping

Then, we do draping on the mesh for each sidescan file, which allows us to get corresponding bathymetry and sidescan based on location. To drape the measurements onto the multibeam mesh, we trace imaginary rays through the water. The computed travel times of the rays are matched to the actual sidescan beams. After we obtain the draping results, we slice the corresponding bathymetry and sidescan into images of size 256×256. Till now, we have obtained the basic dataset.



(a)

(b)

Figure 4.4: Low-resolution bathymetry of different interval factors

Low-Resolution Bathymetry Generation. After generating the basic dataset, we generate low-resolution bathymetry from high-resolution bathymetry. We set a parameter called interval factor to control the interval between beams in low-resolution bathymetry. The larger the interval factor, the wider the interval between every two beams. We generated low-resolution bathymetry with interval factors ranging from 2 to 64 from the high-resolution bathymetry for each data sample. Our model will learn how to use this image as a basis to reconstruct the high-resolution bathymetry by extracting information from the sidescan.

Data Processing. After we get the sliced data, we filter out low-quality images. We convert the sliced images into grayscale images and then normalize the bathymetry and sidescan images, respectively. We find the global maxima and minima pixel values and then normalize the pixel values of all images from 0 to 1. The average pixel value is calculated for each image, and images that are too dark (which means the position is too deep, making the details of the underwater terrain to be difficult to see) are filtered out. We also removed images taken when the AUV is turning, as this may cause undesired deformations.

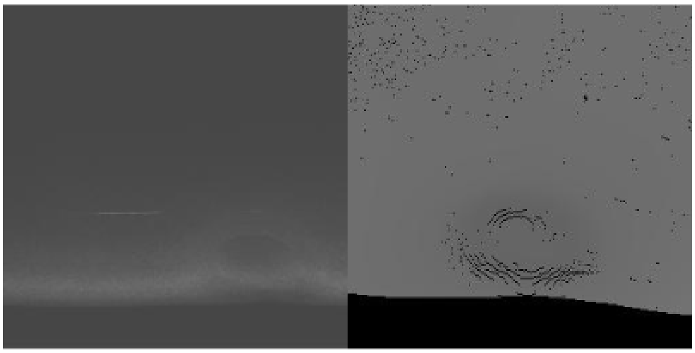


Figure 4.5: A sample in the dataset (A sample that was kept after data processing (Left: the sidescan; Right: the high-resolution bathymetry)

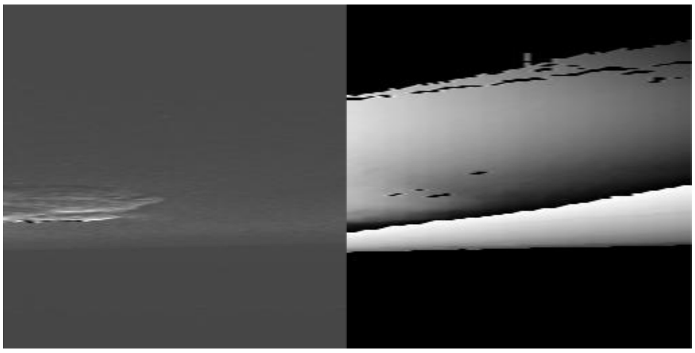


Figure 4.6: A sample in the dataset (A sample that was discarded after data processing, due to the lack of details and low average brightness (Left: the sidescan; Right: the high-resolution bathymetry)

Then, we perform data augmentation to make the dataset more balanced. We flipped 20% of the images (both bathymetry and sidescan). We added Gaussian noise to 25% of the images (only bathymetry).

Chapter 5

Results

5.1 Performance of the Multi-Resolution Inference System

To answer RQ1, we evaluate the output predictions of the system. We compare the output predictions of the system with the predictions generated by the interpolation-based methods. Table 5.1 summarizes the PSNR, SSIM, and MSE scores of different models on the test set.

Inference Methods	**PSNR**	**SSIM**	**MSE**
Nearest Neighbor Interpolation	33.4693	0.9560	0.0010
Bilinear Interpolation	34.1338	0.9531	0.0010
Bicubic Interpolation	33.1040	0.9495	0.0009
Multi-Resolution Inference*	38.5328	0.9626	0.0010

Table 5.1: Comparison of the reconstruction performance using different inference methods

When PSNR is used as the evaluation metric, we see that the multi-resolution inference system outperforms the interpolation-based method. The PSNR score of the multi-resolution inference system (38.5328) is 12.9% higher than that of the interpolation-based method (the bilinear interpolation achieves the highest score of 34.1338). When SSIM is used as the evaluation metric, the performance of the multi-resolution inference system is slightly better, which is 0.7% higher than that of the interpolation-based method (the nearest neighbor interpolation achieves the highest score of 0.9560). When MSE is used as the evaluation metric, all models perform equally well (0.0010).

5.2 Effect of the Input Bathymetry Resolution

To answer RQ2, we measure the influence of different input bathymetry resolutions on the inference performance of the system. Again, we use PSNR as the evaluation metric. We control the training parameters to be the same, as shown in Table 4.1. We use low-resolution bathymetry with different interval factors (as indicated by Figure 4.4) as input for each experiment. These inputs will be passed into the inference system after interpolation, together with the sidescan. We record the PSNR scores obtained by the inference model under different settings in Table 5.2. We also measure the performance of the inference-based method (we choose the best performing method discussed in Section 5.1, which is the bilinear interpolation) on the multi-resolution inference task to facilitate comparison of different methods.

Interval Factors	Inference PSNR	Interpolation PSNR
2	38.5328	34.1338
4	35.1967	30.8787
8	32.7931	28.6271
16	31.1992	26.6550
32	28.9413	24.5376
64	25.5100	21.5334

Table 5.2: Comparison of the reconstruction performance with different interval factors

From the data obtained, we observe that the multi-resolution inference system outperforms the inference-based method for all different interval factors. When interval factors range from 2 to 64, the multi-resolution inference system is on average 4.3% higher than the inference-based method. When the value of the interval factor is 16, the difference in the PSNR score reaches the maximum (4.6% than the inference-based method). When the value of the interval factor is 64, the performance of the multi-resolution inference system and the inference-based method is the closest, with a difference of 3.7%. The results show that the multi-resolution inference system can improve the performance in the multi-resolution inference task.

5.3 Efficiency of the Multi-Resolution Inference System

To answer RQ3, we evaluate the efficiency of the multi-resolution inference system. We use all the data in the multi-resolution dataset for evaluating the pipeline efficiency. We recorded and compared the run time and memory usage of the system. The results are shown in Table 5.3.

Inference Methods	**Time (ms)**	**Memory Usage (MiB)**
Interpolation	3	5
Inference	1823	5243
Total	1826	/

Table 5.3: Time cost and memory usage of different modules of the multi-resolution inference system

Run Time. Since the input images are of the same size, the inference time of the system for different samples is very close. The average running time of the entire system is 1826ms. Among them, the interpolation of the input low-resolution bathymetry takes an average of 3ms, and the generation of the complete high-resolution bathymetry from the interpolated bathymetry takes 1823ms.

Memory Usage. Interpolation of the low-resolution bathymetry by the system takes up 5MiB of memory. When the system generates inference, it takes up an average of 5243MiB of memory. More memory is used during the inference because the system needs to load a large neural network model with a massive number of parameters. It should be noted that if we change the size of the neural network, the memory occupied by the system during inference will change accordingly. At the same time, the memory occupied by the system during the inference is positively related to the batch size.

Chapter 6

Limitations

We identify several limitations caused by the current design choices in the multi-resolution inference model.

First, our model can only be used for fixed-depth high-resolution bathymetry image reconstruction. When we use datasets with different depths (for example, a mix of 30m to 70m) for training, we observe a decrease in the model's performance. Additionally, when we evaluate the model trained on a dataset of a certain depth (e.g., 50m) using a dataset of a different depth (e.g., 30m or 70m), the performance also decreases significantly. We believe that this is because the sample size in the dataset is not large enough. In the future, the model will likely be able to support image reconstruction at different depths. One solution is to expand the number of samples at each depth in the dataset. Another option is to feed depth into the model as an additional parameter, allowing the model to encode depth features explicitly.

Second, our model can only be applied in limited types of conditions. Although our dataset contains images of different underwater conditions, many samples of different special scenarios are still missing. For example, different water temperatures may cause different sound speeds, in which circumstances we need to use different parameters for draping for processing the sidescan images. For another example, the turbid water surface will cause bathymetry to produce different brightness than normal and will produce larger noise in the image. In addition to these two cases, differences in the geology of the seafloor, differences in undulation conditions, and possible obstacles in the water can also significantly affect the performance of our model.

Third, the AUV may have attitude, position, and speed changes when sailing. Currently, our model cannot distinguish this case. When this happens, the detectors may collect incomplete or even erroneous information. For example, when the vehicle is navigating into a corner and needs to make a turn, the outside of the curve is faster than the inside of the curve, so the density of pings emitted and received by the sidescan sonar system is more sparse on the outside of the curve. Conversely, the pings on the inside of the corner will be denser. Coincidence may even occur when the angle of the turn is large. These issues are attempted to be corrected when we preprocess the dataset. Usually, when we do draping, we delete the samples taken by the detectors as the AUV ascends, descends, and turns. However, during a cruise at a constant speed, the AUV may still experience slight steering and depth/speed changes due to current disturbances. A possible solution would be to feed the real-time depth, velocity, and heading information recorded by the AUV into the model as additional parameters, allowing the model to explicitly encode the state of the AUV, thus helping the model more accurately reconstruct the high-resolution bathymetry image from the sidescan.

Overall, this project makes a simple attempt to reconstruct high-resolution bathymetry using the sidescan. Although there is still much room for improvement in the model design, our comparative experiments still demonstrate that the sidescan can be helpful in reconstructing high-resolution bathymetry image.

Chapter 7

Discussion

7.1 Discussion of the Results

7.1.1 System Performance

As we discussed in Section 6, we observe a decrease in the model's performance if we use samples of different depths for training instead of using samples of a fixed depth. This is probably because the backbone design does not have the generalization ability to scale the features. Introducing self-attention and position encoding may help to deal with the problem.

In calculating the three scores, we use 0 to 1 as the range of pixel values. Another way is to use 0 to 255 as the range of pixel values. This would lead to a larger variance and a different sign in the PSNR score. However, this would not affect the comparison as long as we make our experiment consistent. As shown in table 5.1, the three metrics have a consistent evaluation of the performance of the system.

7.1.2 The Effect of Interval Factors

As the results shown in Section 5.1, we can see that the PSNR score achieved by the multi-resolution inference system on the multi-resolution inference task decreases as the interval factor becomes larger. Every time the interval factor is doubled, the PSNR score decreases by an average of 2.7%. This means that the interval factor will influence the performance of the system on multi-resolution inference tasks. This infers that the lower the resolution of the input bathymetry, the more difficult it is for the multi-resolution inference system to re-construct the bathymetry accurately. This may be due to the small number of layers of the model, making it more difficult to extract high-dimensional features when the input is sparse. Increasing the number of residual blocks (e.g., 18 or 50) would address this issue.

7.2 Future Work

7.2.1 Improvement

Input Additional Information. As mentioned before, a potentially effective way to improve model performance is to make additional information available to the input. Therefore, we propose the following two types of extra information: Add the navigation information of the AUV, including the attitude, speed, and position of the AUV. This kind of information can help us find possible errors and correct them within limits allowed by the hardware. As mentioned above, if the AUV turns, the sensor's beam density on the left and right sides will be inconsistent. If we add this information to the model's input, we can enable the model to automatically correct for image distortions caused by AUV turns. This function may be possible when the data includes enough samples collected at turnings. If the AUV yaws left and right or floats up and down due to turbulence, the input AUV's position information can also help us fix the deformation when reconstructing the image. On the other hand, add environmental information such as depth, speed of sound in water, and light conditions. Depth information is essential for image reconstruction. Depth information can help us to normalize the samples of the dataset so as to avoid the inconsistency of the data due to the rise or fall of the AUV. In addition, inputting depth can help the image to better recover the texture of the underwater surface. Input depth may also help the model distinguish common objects by size. If the speed of sound in the water is different, we need to use different parameters to do the draping for the sidescan on the mesh. Inputting the speed of sound in water as a model parameter helps us to normalize the dataset. Likewise, entering lighting conditions as parameters into the model helps us to make the bathymetry consistant in the dataset.

Using More Underwater Scenarios for Training. One of the promising improvement plans is to expand the dataset to a wide variety of underwater scenarios. Training with different underwater scenarios can increase the generalization ability of the model. At present, our dataset only contains samples in some locations, and the model cannot accurately distinguish these locations according to the underwater scenarios. Ideally, the model should be able to distinguish between different underwater scenarios. For example, we may need to label which samples are in open areas and which are occluded, in which sample the water is clear, in which sample water is cloudy, and so on. We should also try to ensure that the number of samples in different scenarios is balanced as much as possible. A balanced dataset will help improve the overall performance of the model.

Apply A Better Backbone Network for Feature Extraction. In order to improve the training speed of the model, we currently only use residual blocks with a low number of layers for feature extraction. However, recent research in computer vision has proposed many backbone networks with better comprehensive performance, which can extract richer and higher-dimensional features from images. For example, Vision Transformer (ViT) [13] can better embed pixel-to-pixel relationships. Compared with traditional CNN, the transformer-based neural network can better capture global feature information. In addition, ViT preserves more spatial information than ResNet [10]. This will undoubtedly be of great help for the reconstruction of bathymetry images. ViT is also able to learn high-quality intermediate features, which can help improve the final performance of the model. These advantages of ViT may make it a better choice than ResNet (or a set of residual blocks) to achieve better results on our model.

7.2.2 Extended Applications

Image Completion. Our original design of the inference model can be extended to image completion. So far, our model only considers inputting sidescan and low-resolution bathymetry at the same time to reconstruct the high-resolution bathymetry. However, in practical applications, we are likely to encounter situations where one type of input is missing. The model should guarantee robustness in this case. When low-resolution bathymetry is missing, we should be able to reconstruct the image with the full information contained in the sidescan, albeit with some loss of accuracy. When the sidescan is missing, we may not be able to reconstruct the image with factual information accurately, but we can still make general predictions on the interpolated low-resolution bathymetry, which provides a higher-resolution reconstructed image. Image completion in the case of missing information can help us generate smoother results when we want to perform image reconstruction on a large amount of data.

Real-time Multi-Resolution Video Inference. When our model is fast enough to process the image, we can apply the image reconstruction in real time. In particular, we can apply image reconstruction to the video. The easiest way is to extract discontinuous frames from the video at certain intervals and input them into the model, thereby outputting a series of continuous high-resolution image reconstruction results. In addition, our current model takes a single frame of the image as the model input. In the future, we can consider inputting multiple frames of images at the same time to establish a temporal as well as a spatial relationship between the images, thereby improving the performance of model reconstruction. A single-frame image may lose some information. For example, when there is a seamount in front of the AUV, the detector cannot see behind the seamount. The terrain details behind the mountains are not available to AUVs. However, when the AUV sails forward until the front of the seamount, we can see the whole picture of the seamount. The same applies to small hills. Therefore, if we can get the input images to be connected, we have the opportunity to obtain a more complete reconstructed image. To achieve this, we must have the support of a larger dataset with corresponding annotations.

Chapter 8

Conclusions

In this project, we study how to accurately and efficiently reconstruct high-resolution bathymetry. There are two main challenges in designing such a system: (1) We need to make full use of the information contained in the sidescan; (2) We do not have ready-made datasets. To address these two challenges, we design a multi-resolution inference system that can extract features in the sidescan and rely on these features to reconstruct high-resolution bathymetry on low-resolution bathymetry and build a dataset for system evaluation and neural network training, which is made from the sidescan and the MBES data provided by SMaRC. Comprehensive evaluations are performed of the multi-resolution inference system on the multi-resolution dataset from the aspects of accuracy and efficiency. From the results, we draw the following conclusions: (1) multi-resolution inference system outperforms interpolation-based methods; (2) multi-resolution inference system achieves better performance for various interval factors; (3) multi-resolution inference system is efficient with regard to average run time and memory usage. We conclude that our system can reconstruct high-resolution bathymetry well based on the experimental results.

Bibliography

[1] Kerry Commander and Daniel Sternlicht. “Pioneers in side scan sonar: Julius Hageman and the shadowgraph”. In: *Journal of the Acoustical Society of America* 137 (Apr. 2015), pp. 2307–2307. DOI: 10.1121/1.4920429.

[2] Yiping Xie, Nils Bore, and John Folkesson. “Inferring depth contours from sidescan sonar using convolutional neural nets”. In: *IET Radar, Sonar & Navigation* 14.2 (2020), pp. 328–334. DOI: https://doi.org/10.1049/iet-rsn.2019.0428.

[3] Karl Pearson F.R.S. “LIII. On lines and planes of closest fit to systems of points in space”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. “Latent Dirichlet Allocation”. In: 3.null (2003), pp. 993–1022.

[5] Mark Davison and Stephen Sireci. “Multidimensional Scaling”. In: (Oct. 2012). DOI: 10.1016/B978-012691360-6/50013-6.

[6] Daniel Olsson, Pando Georgiev, and Panos Pardalos. “Kernel Principal Component Analysis: Applications, Implementation and Comparison”. In: vol. 59. Jan. 2013, pp. 127–148. DOI: 10.1007/978-1-4614-8588-9_9.

[7] Pradip Dhal and Chandrashekhar Azad. “A Comprehensive Survey on Feature Selection in the Various Fields of Machine Learning”. In: 52.4 (2022), pp. 4543–4581. ISSN: 0924-669X. URL: https://doi.org/10.1007/s10489-021-02550-9.

[8] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *arXiv 1409.1556* (Sept. 2014).

[9] Christian Szegedy et al. “Going Deeper with Convolutions”. In: *CoRR* abs/1409.4842 (2014). arXiv: `1409.4842`. URL: `http://arxiv.org/abs/1409.4842`.

[10] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *arXiv preprint arXiv:1512.03385* (2015).

[11] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. “Densely Connected Convolutional Networks”. In: *CoRR* abs/1608.06993 (2016). arXiv: `1608.06993`. URL: `http://arxiv.org/abs/1608.06993`.

[12] Mingxing Tan and Quoc Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: (May 2019).

[13] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *CoRR* abs/2010.11929 (2020). arXiv: `2010.11929`. URL: `https://arxiv.org/abs/2010.11929`.

[14] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: (Feb. 2015).

[15] Roger Y. Tsai and Thomas S. Huang. “Multiframe image restoration and registration”. In: 1984.

[16] Amir Pasha Mahmoudzadeh and Nasser Kashou. “Interpolation-based super-resolution reconstruction: Effects of slice thickness”. In: *Journal of Medical Imaging* 1 (Dec. 2014). DOI: `10.1117/1.JMI.1.3.034007`.

[17] Syed Muhammad Arsalan Bashir, Yi Wang, and Mahrukh Khan. “A Comprehensive Review of Deep Learning-based Single Image Super-resolution”. In: *CoRR* abs/2102.09351 (2021). arXiv: `2102.09351`. URL: `https://arxiv.org/abs/2102.09351`.

[18] Chao Dong et al. *Image Super-Resolution Using Deep Convolutional Networks*. 2015. DOI: `10.48550/ARXIV.1501.00092`. URL: `https://arxiv.org/abs/1501.00092`.

[19] Nils Bore. *auvlib*. `https://github.com/nilsbore/auvlib`.

[20] R. Collobert, K. Kavukcuoglu, and C. Farabet. “Torch7: A Matlab-like Environment for Machine Learning”. In: *BigLearn, NIPS Workshop*. 2011.

[21] Alain Horé and Djemel Ziou. “Image Quality Metrics: PSNR vs. SSIM”. In: *2010 20th International Conference on Pattern Recognition*. 2010, pp. 2366–2369. DOI: 10.1109/ICPR.2010.579.

[22] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

Appendix A

Evaluation of Design Alternatives

In this section, we introduce the different neural network architectures we designed and evaluate their performance. The performance of each design on the test set is shown in the following table:

Design	**PSNR**
Design 1	19.1369
Design 2	33.4144
Design 3	33.1237
Design 4	35.9156
Multi-Resolution Inference*	38.5328

Table A.1: Comparison of the reconstruction performance using different neural network design

A.0.1 Design 1

In theory, we can construct high-resolution bathymetry directly from scratch. Here, we demonstrate the simplest model.

Architecture

In this model, we only use sidescan as input. The input will go through two residual blocks, and the output will be the generated high-resolution bathymetry of the same size. The idea is that the sidescan contains all the information to construct the high-resolution bathymetry. The structure of this model is shown in the figure below.

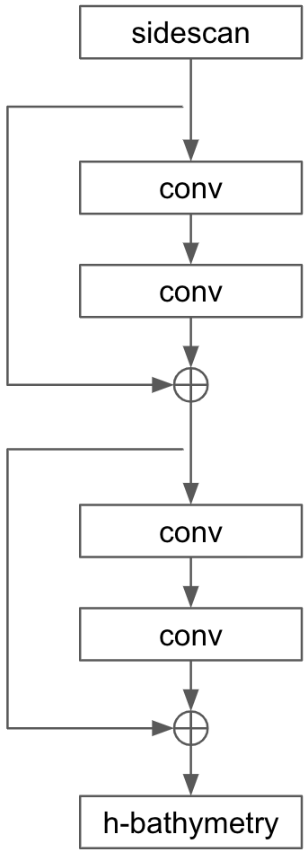


Figure A.1: Architecture of design 1

Discussion

Obviously, this is not the best design. While the sidescan contains more information than the low-resolution bathymetry, the reconstruction accuracy is not good. This is because the reconstructed image and the ground truth image are not well aligned. On the one hand, the difference between input and output is too significant. Therefore the learning process is difficult to converge. Although there is a corresponding relationship between bathymetry and sidescan in the dataset, they are recorded from two different sensors, so there will be a certain deviation in the perspective of viewing the seabed. As we cannot build a pixel-level match from the output and the ground truth, the evaluation results are unsatisfactory.

A.0.2 Design 2

We introduce low-resolution bathymetry to address the alignment issue in the previous model. Using it as an input to the model, we manually introduced the bathymetry's positional features into the model, replacing the original shortcut in the residual blocks. This allows the model to improve its alignment capabilities.

Architecture

In this model, we use sidescan as the main input for feature extraction and low-resolution bathymetry as the auxiliary input to help the model output align. The input will be extracted by a set of residual blocks. The extracted features are passed through additional convolutional layers to generate high-resolution bathymetry. Note that after each residual block, we add the interpolated low-resolution bathymetry to the intermediate result recursively. In this way, we introduce positional features to the feature extraction of each layer.

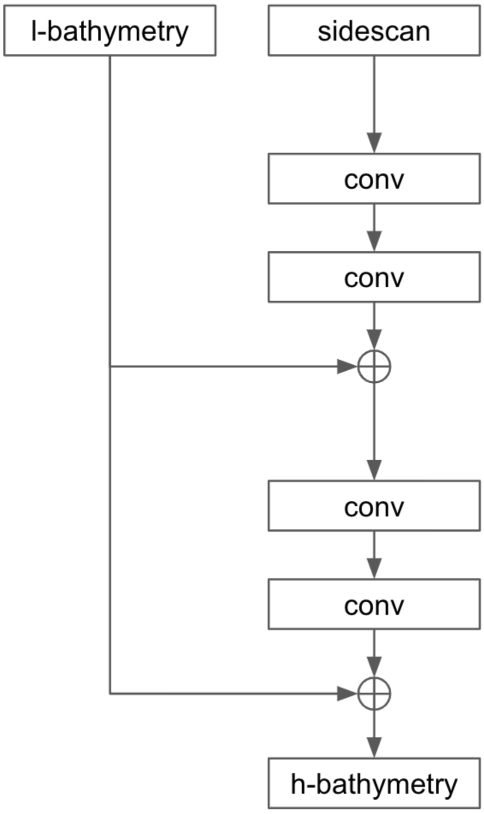


Figure A.2: Architecture of design 2

Discussion

There are also some problems with this design. Although passing in low-resolution bathymetry allows the image to be well aligned, passing it multiple times can also affect the model. This effect can lead to poor feature extraction ability of the model. The behavior is that the output of the model will be highly similar to low-resolution bathymetry. There is no way for the model to effectively recover details using sidescan.

A.0.3 Design 3

In this model, we also use sidescan as the main input and low-resolution bathymetry as the auxiliary input. The difference from the previous model is that we only add low-resolution bathymetry to the last result to let the neural network learn the residual between input and output. In this way, the model implicitly takes positional features into account during the learning process. At the same time, since the feature extraction is independent, the input of low-resolution bathymetry does not interfere with the extraction process.

Architecture

Like the previous models, we use multiple residual blocks for feature extraction on the sidescan. For each residual block, we apply the standard shortcut design to connect the input and output. In this way, the model will learn the residual between the features. The training speed can be improved, and the problem of network degradation can be effectively solved. The difference from the previous models is that we add the interpolated low-resolution bathymetry to the results reconstructed from the features. In this way, we let the network learn the residual between low-resolution bathymetry and high-resolution bathymetry. This not only allows the model to align positionally during the learning process automatically but also speeds up the learning process.

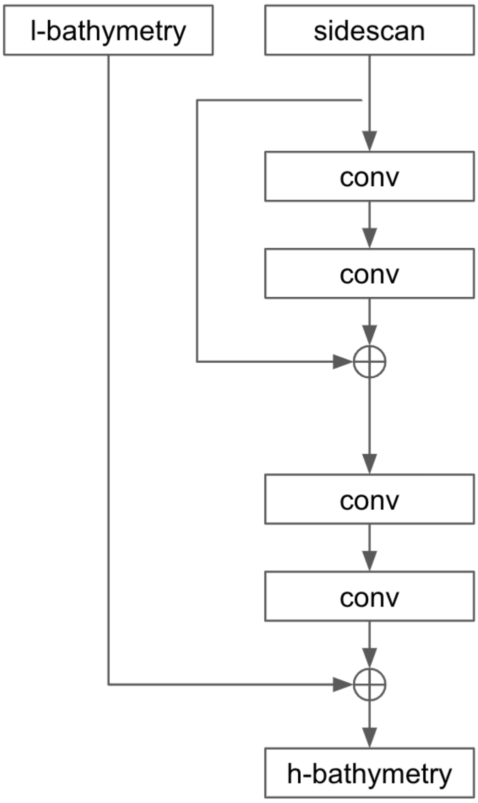


Figure A.3: Architecture of design 3

Discussion

Compared to the previous model, the design of this model is more reasonable. Sidescan, which is the main source of information, is used for feature extraction, while low-resolution bathymetry, which has higher accuracy, is used as the reference image for alignment. The network only learns the residual between the reference image and the reconstructed image in the last step, improving the speed of training. Since the process does not affect feature extraction, it ensures the accuracy of the obtained results. However, this model still has room for improvement because low-resolution bathymetry is not fully utilized.

A.0.4 Design 4

In this model, we use a more complex structure for feature extraction. We hope to use accurate information from low-resolution bathymetry to help achieve high-quality reconstructions. We perform feature extraction on low-resolution bathymetry and sidescan separately and jointly feed these features into another set of convolutional networks to generate high-resolution bathymetry.

Architecture

We use the same structured model for feature extraction for low-resolution bathymetry and sidescan. In each feature extractor, we use multiple sets of residual blocks for feature extraction to obtain rich feature expressions. Similarly, for each residual block, we apply the standard shortcut design to connect the input and output. After we extract the features of low-resolution bathymetry and sidescan, respectively, we concatenate the obtained larger set of feature maps. We can do this because both feature extractors output the same size and number of feature maps. Then, we pass the concatenated feature maps to another set of convolutional neural networks to generate high-resolution bathymetry. It should be noted that to verify the effectiveness of the feature extractor design, we did not add a shortcut in the final step.

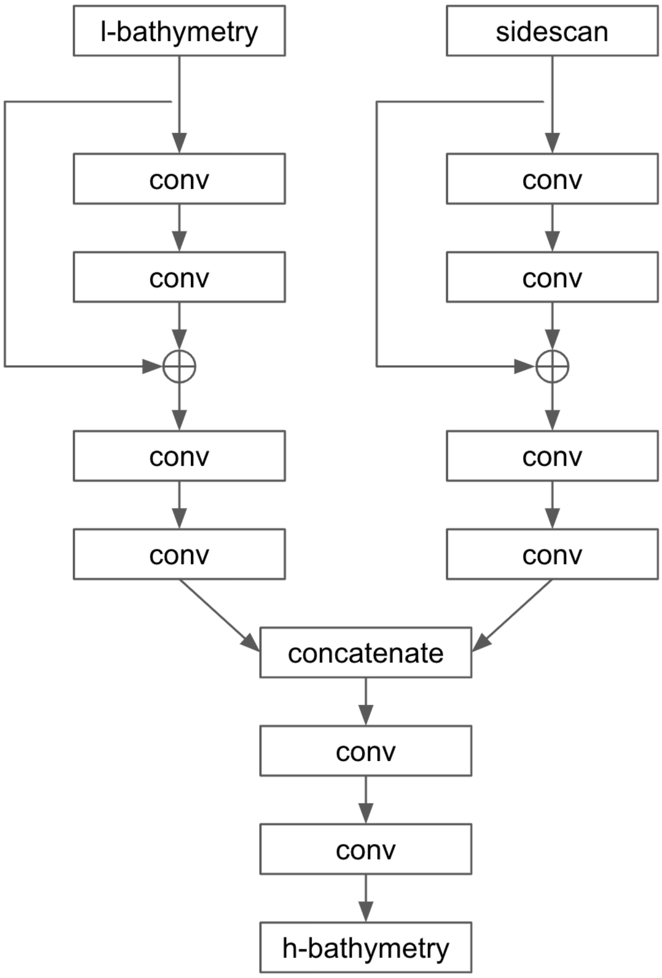


Figure A.4: Architecture of design 4

Discussion

Compared with the previous model, we fully use the pixel location information in the low-resolution bathymetry. By using two sets of residual blocks to extract information in low-resolution bathymetry and the sidescan, respectively, the model is not only accurate but also performs well in locating the features. Since experiments in design 1 and design 2 demonstrate that a shortcut between the input of interpolated low-resolution bathymetry and output of high-resolution bathymetry can improve the model performance, we finally proposed our final design of the multi-resolution inference neural network model, as shown in Figure 3.2. Notice that comparative experiments show that the model performs better using interpolated low-resolution bathymetry than passing low-resolution bathymetry directly. A detailed description of the model is provided in 3.3.

TRITA-EECS-EX-2023:90