Ocean Floor Contour Prediction

Merve Kilic
ECE
UC San Diego
La Jolla, CA
mkilic@ucsd.edu

Maxime Ghesquiere
ECE
UC San Diego
La Jolla, CA
mghesqui@ucsd.edu

Mike Liu
ECE
UC San Diego
La Jolla, CA
m7liu@ucsd.edu

Abstract

The ocean floor is a complex and ever-changing landscape that is difficult to explore due to the high pressure and darkness of the deep sea. Mapping the ocean floor is crucial for various applications such as navigation, marine resources management, and predicting natural hazards. Traditional methods of ocean floor mapping involve expensive and time-consuming ship-borne surveys, which can only cover a small portion of the ocean floor. In this paper, we explore machine learning models for predicting ocean floor contours. Using data from the NOAA National Centers for Environmental Information (NCEI) website, we compare linear regression, convolutional neural network (CNN), pre-trained U-Net, and conditional deep convolutional generative adversarial network (DCGAN) models using mean squared error and structural similarity metrics. While most of the models produce highly noisy results, the pre-trained U-Net model achieves great performance, accurately predicting contours and offering potential cost and time savings in seafloor mapping.

1 Introduction

1.1 Background

The earth is 70% covered by the ocean and less than 20% has been fully mapped, explored, and observed. This situation is attributed to the same electromagnetic absorption qualities of water that made life on Earth possible, rendering detailed seafloor mapping difficult. Electromagnetic waves in all spectra reach at most 100m depth in the ocean where the average depth is 3688m. [Fig.1.1] As a result, most satellite and remote sensing techniques are impossible to describe the ocean floor accurately.

To achieve any reasonable level of accuracy, modern bathymetry relies on the Doppler effect of acoustic signals propagating through the water. Because acoustic signals require a sufficiently dense medium to propagate, where the denser the medium the stronger the acoustic return. In effect, the high density of water allows for high-accuracy mapping that is not possible with EM waves. The most modern bathymetry methods rely on Multi Beam Echosounder (MBES) mounted on the bottom of ships, then run in strip mine patterns across unmapped areas. However, the cost of operation is prohibitively expensive and gaps would still remain due to the strip mining pattern. For example, to map a 100km^2 area with 500m tracks at 7.4km/h speed on the Robert E. Sproul [Table 1] would require $(100,000 * 2/7.4 = 2702.07h)$ and in excess of $(2702.07\ 24 * 15,000 = \$1,689,189.19)$. Not accounting for travel time from port to destination and return. For reference, Disney World in Florida is 100km^2.

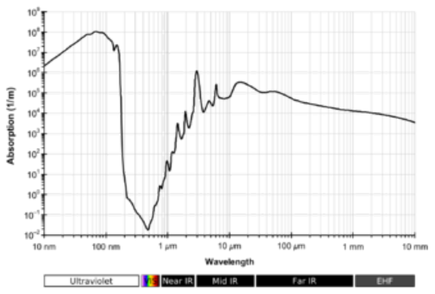

Figure 1: EM wave penetration depth in seawater

Table 1: Typical Research Vessel Costs

Ship name	Price(USD)	Ship Type
Sproul	$15,000	Regional
Sally Ride	$45,000	Ocean
Revelle, Meteor, Atalante	$60,000	Global
Polarstern	$90.000/day	Ocean

1.2 Motivation

Hence, the high costs of ocean bathymetry have made full mapping of the ocean floor an almost impossible task. Instead, most ocean floor data available are low-quality contours derived from slight height differences of large underwater structures reflected on the ocean surface. However detailed information such as sand banks, miniature trenches, and wrecks are difficult to estimate due to their small mass. We hypothesize that by using machine learning algorithms it would be possible to fill in the gaps and predict regular trends on the ocean floor.

2 Related Work

There are very few publicly related works on ocean contour prediction. Most oceanography prefers in situ experiments and monitoring, so contour prediction is a feature more prevalent in medical and astronomy applications. BathyNet uses Unet for Water Depth Mapping from Multispectral Aerial Images specific to lakes, dams, and rivers. GEBCO is part of the UN Seafloor 2030 in effort to completely map the seafloor by 2030. Google Ocean collated global altimeter data with geoid models and gravity data.

3 Methodology

3.1 Dataset

3.1.1 Data Collection

The dataset is pulled from the National Oceanic and Atmospheric Administration (NOAA) bathymetry data collated from all vessels affiliated with NOAA equipped with a multibeam scanner. Since most of the ocean is not well-mapped, the obtained datasets are concentrated in hotspots most interesting to Oceanography and the US Navy. Namely, the Hawaiian islands, North American East and West Coast, sections of Mid Atlantic Ridge, Mariana Trench, and parts of surrounding waters near Papua New Guinea. The data is then downloaded as a .TIF file with rectangular sizes manually inputted into the NOAA maps through longitude and latitude coordinates. As part of the metadata, we saved our most Northern and Western parts of each .TIF file and our core values are from the ocean surface of 0 meters and the deepest depth in the Mariana trench of -11,000 meters.

3.1.2 Data Preprocessing

The collected images have varying height and width, making it hard to pass them directly into machine learning algorithms. In order to standardize our data, we split each image into a grid of 100pix by 100pix squares. Each square is considered a training datapoint. The input to the networks is an 80pix by 80pix center crop, and the ground truth is the full square image.

The data also contains NaN values in locations that have not been surveyed. We deal with these NaNs by replacing them with the mean of the non-NaN values in the ground truth. Some samples have only NaN values, these are thrown out as unusable.

3.2 Models

In order to predict ocean floor contours, we set up our model so that we can predict the spread of an existing area in a supervised manner. Thus, our input was the center section of our ground truth image. Because of this direct translation of the input image as the center of the output image, we chose to experiment with a U-Net model [2]. Additionally, we chose to experiment with a conditional GAN model to see whether it could generate realistic new contours from an existing area.

Next, in order to have a baseline for comparison of our models' performances, we chose two baseline models. First, we chose linear regression as a straightforward, efficient test. Then, we chose to do a convolutional neural network (CNN) model as our second baseline to have a deep learning based baseline.

In order to measure our model's performance, we chose two metrics. We used mean squared error (MSE) as the loss function while training our models so that we would know how different the values of our generated contour image matrix were from the ground truth image matrix. We also chose to measure the structural similarity index (SSIM) between the predicted and ground truth images so that even if the image is shifted, the overall shapes and structure could be evaluated rather than each value directly.

We split our dataset into a training and testing set, using 70% for training and 30% for testing. We evaluated the MSE and SSIM scores on the testing sets for use for comparison.

3.2.1 Linear Regression

Our linear regression model was very straightforward. We used Scikit-learn's standard linear regression model.

3.2.2 Convolutional Neural Network

Our CNN model was a simple five-layer model. The first three layers were convolutional. We used a kernel size of 3, a stride of 1, and a padding of 1. Then, we used batch normalization and the reLU activation function after each layer. Next, we flattened our output and had a fully connected layer. Lastly, another fully connected layer changed the output size to the one we desired, 10,000, due to our output shape being 100x100.

First, we trained our model without batch normalization and with a batch size of 32. Our model only produced noise as output, so we added batch normalization and increased our batch size to 128. We trained for 100 epochs using the Adam optimizer and a learning rate of 0.001.

3.2.3 U-Net

The architecture of U-Net, as shown in figure 2, has skip connections throughout, making it seem like a good fit for our task due to our input image being directly in our output image. U-Net's architecture entails that the input and output be of the same size, so we padded our input image with zeros to get the same size.

Although it was our baseline, after seeing unsatisfying results for our CNN model, we chose to use a U-Net model consisting of a pre-trained encoder. We chose VGG11 [3] pre-trained on ImageNet [1] because it is good at image segmentation which we thought would translate well.

While training, similar to our CNN model, we used a batch size of 128, the Adam optimizer, and a learning rate of 0.001. We also had to perform our test in batches due to memory issues. We trained the U-Net for 100 epochs.

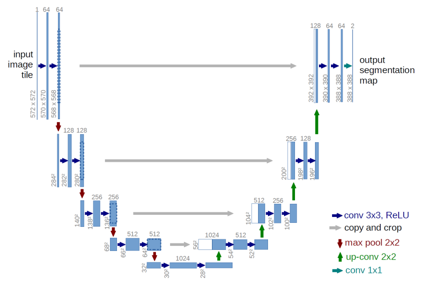

Figure 2: U-Net Architecture [2]

3.2.4 Conditional GAN

Another model we wanted to try was a conditional DCGAN. We wanted to see how it would perform on this task given its success as a generative model for images. The cGAN is composed of an encoder and a generator and a discriminator.

The generator has three 2D convolutional layers sandwiched between 2 fully connected layers. The generator takes in a Gaussian noise vector of length 100, and the input 80x80 center crop, flattened into a vector. It outputs a vector of length 10*90*4 which we can map back to the frame of the ground truth image. This choice was made to reduce memory load.

The discriminator takes in the flattened 80x80 center crop given to the generator, as well as the flattened frame of either the ground truth or the generated frame. This flat vector is fed into a similar structure to the generator, but with the last fully convolutional layer outputting a pair of scores for generated vs true, which we normalize using softmax.

The generator and discriminator are trained against each other with a zero sum GAN loss.

4 Results

The training loss plots of our models are shown in figure 3. Our resulting model performances are found in table 2, showing each model's MSE and SSIM values. In order to visualize our results, we have chosen an image from our dataset to produce predictions from. The input and true output images are found in figure 4 and the model predictions are in figure 5. Our code is provided at `https://drive.google.com/file/d/1GkFGRdiomopsSsiv-kOG6MydgNu-DGiP/view?usp=sharing`

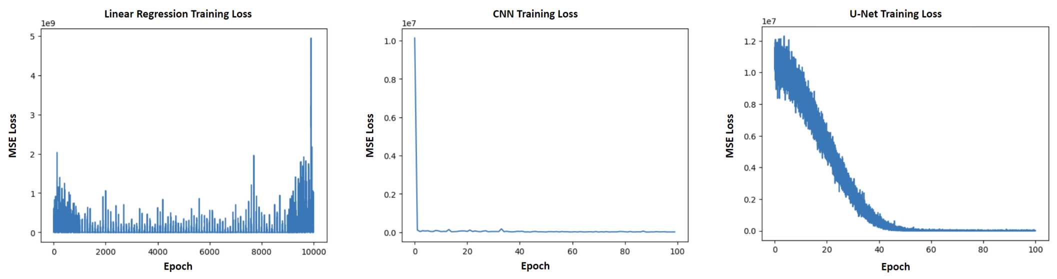

Figure 3: Loss Plots

4.1 Linear Regression

Our linear regression model achieved a MSE of 5781.53 and SSIM of 0.8381. While this SSIM score seems high, the predicted output shown in figure 5 shows something different. The portion of

Table 2: Model Performances

Model	MSE	SSIM
Linear Regression	5781.53	0.8381
CNN	11464.52	0.6955
U-Net	2216.99	0.9848
cGAN	0.3028	0.0174

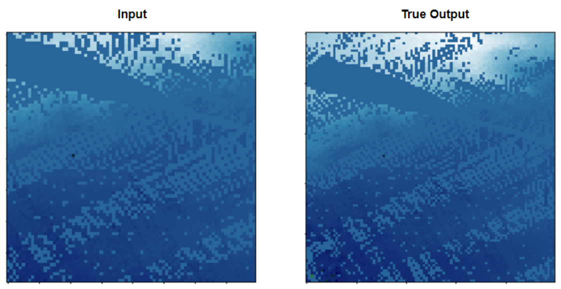

Figure 4: Testing input and true output

the output translated from the input is predicted perfectly while the new portion is predicted to be highly noisy. This is because linear regression can directly copy the center of the image, but cannot predict new parts well. Thus, the SSIM score is high because the center of the image is identical to the ground truth image's center. Still, we see that linear regression was unable to perform well.

4.2 Convolutional Neural Network

Our CNN model achieved a MSE of 11464.52 and SSIM of 0.6955. The predicted image shows a gradient matching the true output image but no shapes, indicating that our CNN model is able to learn low-frequency features but struggles in learning high-frequency features.

4.3 U-Net

Our U-Net model achieved a MSE of 2216.99 and SSIM of 0.9848. These are great scores compared to our baseline, showing that our model performs well. Further analysis of the predicted image shows that besides some blur in the new areas, the prediction matches the true output very well.

4.4 Conditional GAN

We trained the cGAN for 50 epochs with a learning rate of 0.02 and a standard Adam optimizer. For memory reasons, this was done as 5 checkpoints of 10 epochs. The GAN loss plot is not meaningful, as both the generator and discriminator losses stay very near their starting values of 1.4 and 0.7 respectively for the entirety of training. As such, we do not show it to save space. Each checkpoint of 10 epochs took 3 hours to train, and despite this, the results are a little better than simple Gaussian noise [Fig. 5] Note that the MSE seems small since the data is normalized to between 0 and 1 before being passed to the cGAN. The SSIM gives a better approximation of the quality than the MSE. Note that both the MSE and SSIM are calculated using only the 10pixel wide frame, not the center crop.

5 Discussion

When we look at the SSIM scores in Table 2, we see that the U-Net was the most successful at recovering the true outer frame, and the cGAN was by far the least. The MSE numbers here are misleading, as the cGAN uses a normalized input while the others use the raw input with range (-6000,0). While the linear regression has a high SSIM, this is because it is able to find the identity relation between the input and the center crop of the output, qualitative analysis [Fig.5] shows that

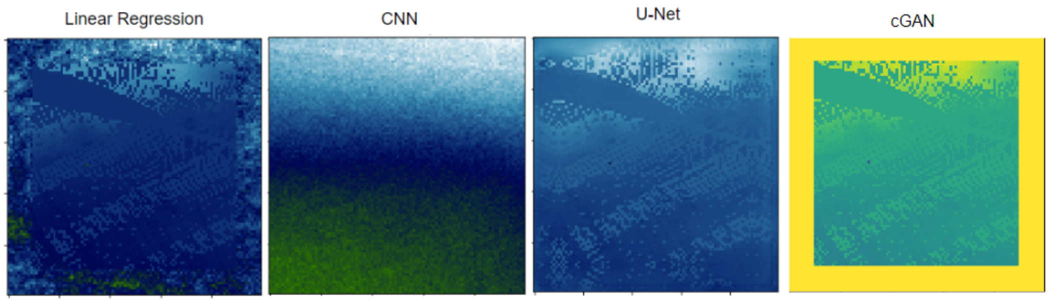

Figure 5: Model Output Contours

the outer frame is very noisy. On the other hand, while the CNN has a comparatively lower SSIM, it predicts the low spatial frequency structure well. If we trained a deeper CNN for longer, it may be able to reach performance similar to U-Net. The cGAN performs terribly both on the SSIM metric and by qualitative analysis, and is therefore not a good choice of model for this task. The loss plots for the first three models [Fig. 3] shows that the CNN and U-Net do converge, while the linear regression loss seems very noisy.

6 Conclusion

Our original intention to use cGAN to predict the ocean contour was a failure for two reasons. First, cGAN inherently takes a massive amount of data, time, and resources to train. We fulfilled the first condition by cutting up our inputs to 100x100 mapping of our original 102 NOAA maps. However, the last two conditions cannot be fulfilled without supercomputer access. We concluded that the cGAN was either not learning anything or learning at a rate so slow all of our outputs are noise. This is in contrast to Linear Regression, CNN, and U-Net, where U-Net was our best performing model. Linear regression and CNN were predicting the anomalies or very rough outlines of underwater structures but still contain too much noise for use. In contrast, U-Net was able to continue to predict the frame of our input relatively well. It was able to extend systematic structures that are incredibly important to oceanography since repeating geography is common in plate tectonics and ocean floor. In conclusion, while we failed in the original goal to use cGAN as a prediction model, we proved that with U-Net it is possible for rudimentary contour prediction of the ocean floor in vastly less time and resources.

7 Individual Contributions

- Merve Kilic:
  Implemented, trained, and ran some experiments for linear regression, CNN, U-Net.
- Maxime Ghesquiere:
  Write code for data cleaning and loading, cGAN, and loss curves. Train cGAN. Helped write code to display results.
- Mike Liu:
  Data collection (manual), helped train CNN, helped write code to display results

References

[1] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs.CV].

[3] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].