ECE:5995 Applied Machine Learning

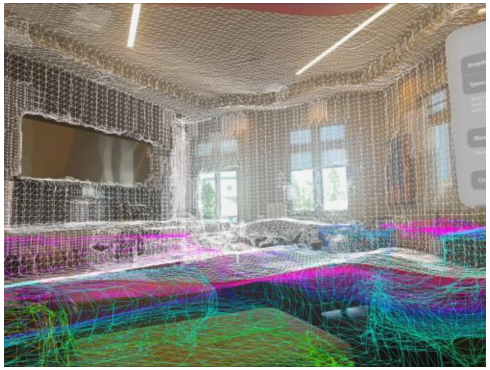
Neural RGB-D Encoding Optimized for Image Codecs

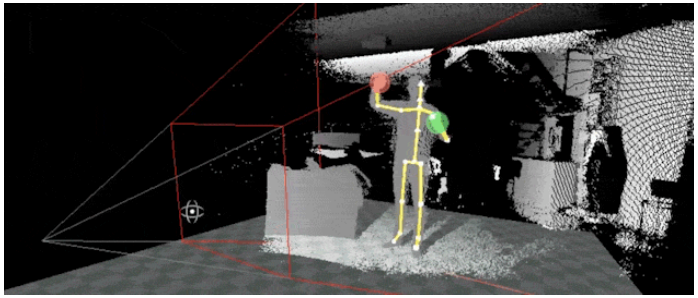
Nick Hageman and Nathan Schaefer

March 30, 2024

Overview

- With technologies like the Apple Vision Pro and personal avatars becoming more accessible, 3D capture technology has seen recent advancements



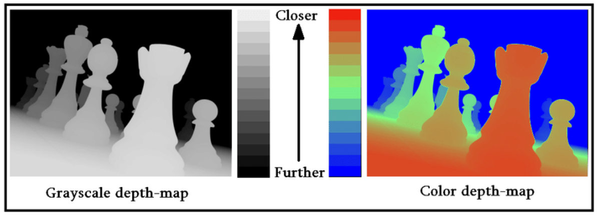


- Transmitting 3D data is expensive, especially on hardware limited devices (mobile, XR headsets, etc.)
- Compression by modern image and video codecs is highly optimized and often hardware accelerated
  - Leverage this by creating an end-to-end neural network sandwiched around a differentiable version of JPEG for our RGB-D encoding scheme

Background of 3D data transmission

Color & Grayscale Depth Maps

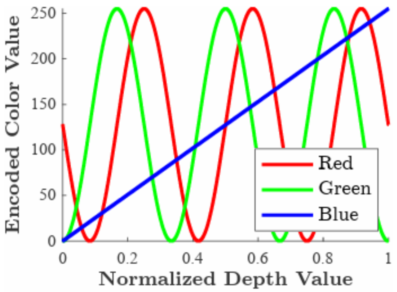
Grayscale maps use shades from black to white to indicate depth, with darker shades showing deeper areas. Color maps assign colors to different depths, making distinctions clearer and easier to understand at a glance.



Multiwavelength Depth Encoding (Holo Reality Lab)

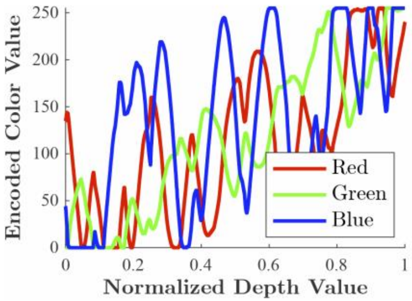
The MWD handcrafted encoding scheme works by storing two sinusoidal encodings and a normalized depth map into the RGB channels of a color image.

[J2] T. Bell and S. Zhang, "Multiwavelength depth encoding method for 3D range geometry compression," Appl. Opt. 54(36), 10684-10691, 2015.

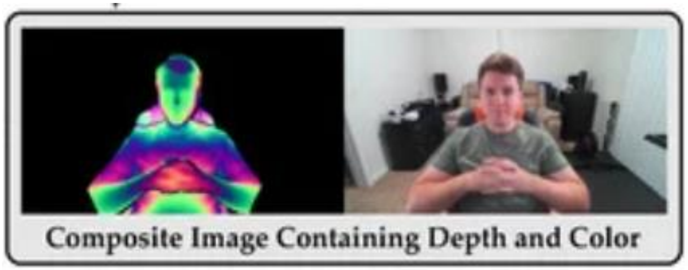


Neural Depth Encoding (Holo Reality Lab)

First depth-to-RGB encoding directly optimized for a lossy image compression codec. Consistently outperformed MWD with compressed image sizes, on average, more than 20% smaller.

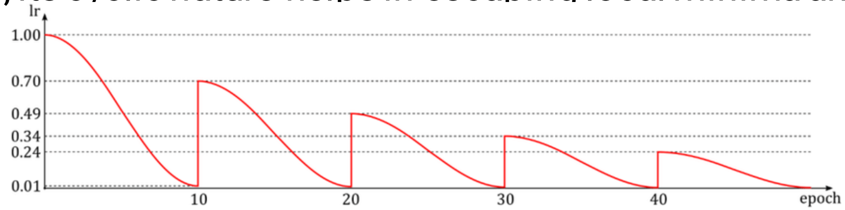


Approach



- Problem
  - Still need to send the associated RGB frame with their encoded depth map side-by-side
  - We want to encode all the information (RGB+D) into one single RGB frame

- Monkaa Dataset
  - ~35,000 stereo frames with ground truth for disparity maps
  - Split into training (20% validation) and testing sets
- Data Augmentation
  - Experimented with adding Gaussian Noise, Color Jitter, and Random Crops
- Training & Validation
  - Saved weights depending on the model's performance on the validation dataset each epoch
  - Used Adam optimizer for its adaptive learning rate
  - Used a cosine annealing scheduler, its cyclic nature helps in escaping local minima and saddle points during training

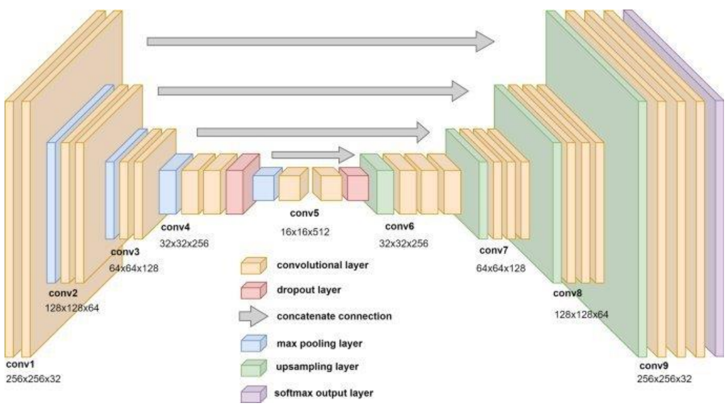


Approach

Constructing our Neural Network

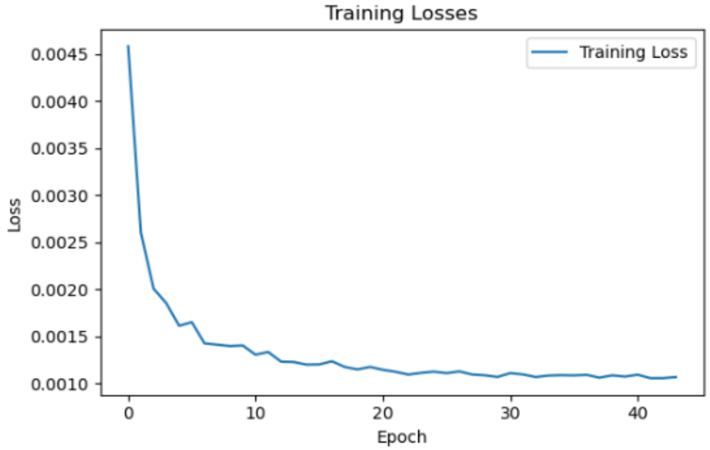
- **U-Net** (Consists of double convolutions, ReLU activations, downsampling, and upsampling layers)
- **MLP** (Consists of 3 layers of 2D convolutions & ReLu activation functions)
- Added the two in parallel, passed through a sigmoid activation function, then passed through differentiable JPEG
- Same structure (parallel U-Net & MLP) for the decoding

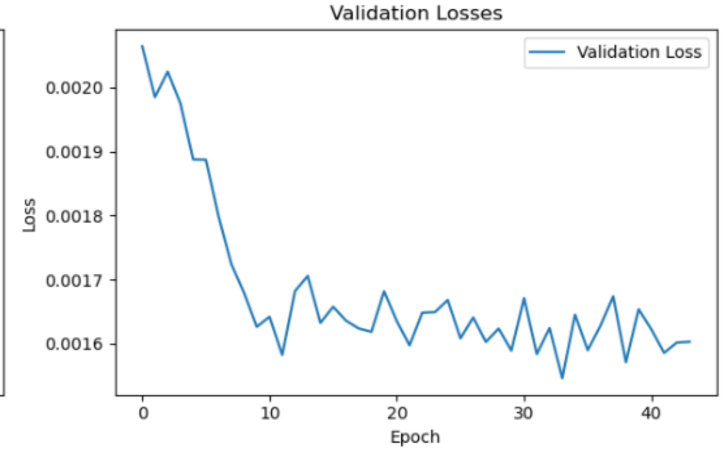
General U-Net structure with skip connections



General structure of our network sandwiched around an image codec

Results





Training & Validation results

- Saw consistent trends in minimizing losses
- Seemed to have found local minima

```
Epoch 0 Training 5066: 100%|██████████| 5066/5066 [12:21<00:00,  6.84it/s, Training Loss: 0.0701]
Validation Training 2004: 100%|██████████| 2004/2004 [01:57<00:00, 16.98it/s, Validation Loss: 0.0021]
Epoch 1 Training 5066: 100%|██████████| 5066/5066 [12:03<00:00,  7.00it/s, Training Loss: 0.0575]
Validation Training 2004: 100%|██████████| 2004/2004 [01:57<00:00, 17.03it/s, Validation Loss: 0.0020]
Epoch 2 Training 5066: 100%|██████████| 5066/5066 [12:05<00:00,  6.98it/s, Training Loss: 0.0527]
Validation Training 2004: 100%|██████████| 2004/2004 [01:57<00:00, 17.05it/s, Validation Loss: 0.0020]
Epoch 3 Training 5066: 100%|██████████| 5066/5066 [12:05<00:00,  6.99it/s, Training Loss: 0.0511]
Validation Training 2004: 100%|██████████| 2004/2004 [01:58<00:00, 16.97it/s, Validation Loss: 0.0020]
Epoch 5 Training 5066: 100%|██████████| 5066/5066 [12:05<00:00,  6.98it/s, Training Loss: 0.0476]
Validation Training 2004: 100%|██████████| 2004/2004 [01:52<00:00, 17.80it/s, Validation Loss: 0.0019]
Epoch 6 Training 5066: 100%|██████████| 5066/5066 [12:06<00:00,  6.97it/s, Training Loss: 0.0462]
Validation Training 2004: 100%|██████████| 2004/2004 [01:57<00:00, 17.00it/s, Validation Loss: 0.0018]
Epoch 8 Training 5066: 100%|██████████| 5066/5066 [12:12<00:00,  6.91it/s, Training Loss: 0.0448]
Validation Training 2004: 100%|██████████| 2004/2004 [01:59<00:00, 16.75it/s, Validation Loss: 0.0017]
Epoch 9 Training 5066: 100%|██████████| 5066/5066 [12:13<00:00,  6.91it/s, Training Loss: 0.0448]
Validation Training 2004: 100%|██████████| 2004/2004 [01:58<00:00, 16.93it/s, Validation Loss: 0.0016]
Epoch 10 Training 5066: 100%|██████████| 5066/5066 [12:19<00:00,  6.85it/s, Training Loss: 0.0439]
Validation Training 2004: 100%|██████████| 2004/2004 [01:56<00:00, 17.23it/s, Validation Loss: 0.0016]
Epoch 11 Training 5066: 100%|██████████| 5066/5066 [12:22<00:00,  6.83it/s, Training Loss: 0.0438]
Validation Training 2004: 100%|██████████| 2004/2004 [01:49<00:00, 18.28it/s, Validation Loss: 0.0016]
Epoch 12 Training 5066: 100%|██████████| 5066/5066 [12:19<00:00,  6.85it/s, Training Loss: 0.0424]
Validation Training 2004: 100%|██████████| 2004/2004 [01:48<00:00, 18.48it/s, Validation Loss: 0.0017]
Epoch 13 Training 5066: 100%|██████████| 5066/5066 [12:19<00:00,  6.85it/s, Training Loss: 0.0427]
Validation Training 2004: 100%|██████████| 2004/2004 [01:55<00:00, 17.29it/s, Validation Loss: 0.0017]
Epoch 14 Training 4836:  95%|█████████ | 4835/5066 [10:54<00:31,  7.40it/s, Training Loss: 0.0428]
```

Testing results

- Produced an MSE error of 0.0010 with unseen data from the test set

```
Testing 1594: 100%|██████████| 1594/1594 [01:36<00:00, 16.50it/s, Testing Loss: 0.0010]
```

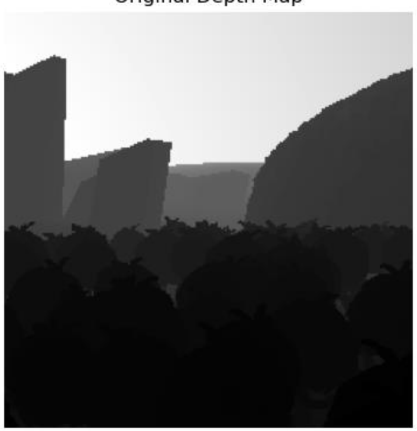
Experimented with several loss functions, ended up going with MSE for our final model

```
MSE_criterion = nn.MSELoss()
L1_criterion = nn.L1Loss()
BCE_criterion = nn.BCEWithLogitsLoss()
```

Results

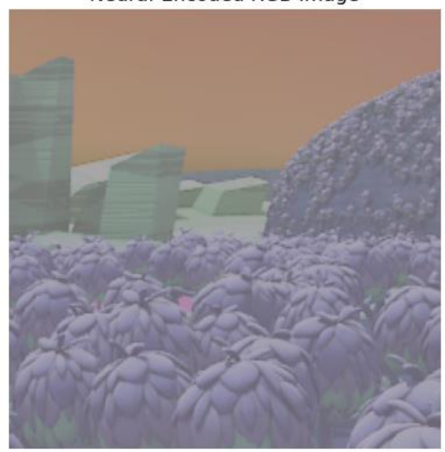
Original RGB Image

Original Depth Map



56 bits/pixel

Neural-Encoded RGB Image



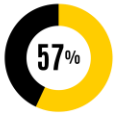
24 bits/pixel

Recovered RGB Image

Recovered Depth Map

56 bits/pixel

Reduced original file sizes by



Conclusion

- Current state of the project
  - Novel approach to encoding RGB-D data
    - Significant opportunities to further develop and enhance our model
  - Our 1st significant image processing project with deep learning
    - Learned a lot about autoencoders & MLPs
- Future work
  - Experiment with varying JPEG qualities
  - Look into using a perceptual loss function for the RGB layers
  - Look into masking the data as 3D data typically has a "range" of values that aren't relevant
    - Could a network take advantage of this unused space for our encoding/decoding scheme?

Thank you

uiowa.edu

Members
Nick Hageman – nicholas-hageman@uiowa.edu
Nate Schaefer – nathan-schaefer@uiowa.edu

Advisors
Stephen Siemonsma – stephen-siemonsma@uiowa.edu
Tyler Bell – tyler-bell@uiowa.edu

Github Repo
github.com/Nick-Hageman/Neural-RGBD-Encoding

Citations

– Multiwavelength depth encoding method for 3D range geometry compression
– Sandwiched Image Compression: WrappingNeural Networks Around A Standard Codec
– PyTorch U-Net implementation GitHub repository