3D Scene reconstruction from 2D images

SRIHARI INUKURTHI, University of Colorado Denver, USA

Medical endoscopy is a standard in the clinical process that enables doctors and medical practitioners to diagnose and analyze various things that happen in a human body. It still remains a challenging task to make better decisions just from the endoscopy images because of the sparsity of the features of the image and size constraints where the depth information is completely lost. For any 2D endoscopic image, there are always a various number of ways that you can try to explain the same 2D image. So, the three dimensional scene must be estimated with respect to the global context. The solution that is proposed in this project uses modern Deep Learning techniques and unsupervised learning algorithms to predict the depth map and then generate the 3D scene using the point cloud. By using the ResNet-18 and Encoder-Decoder architectures, a model was trained on the images that is a combined version of publicly available Hamlyn data and the images taken from endoscopic videos. The model provides a good understanding of the internal endoscopic results, thereby helping in making better decisions.

CCS Concepts: • **Deep Learning, Convolutional neural network, Endoscopy, Scene reconstruction**;

Additional Key Words and Phrases: Encoder-decoder, point cloud, depth map

ACM Reference Format:
Srihari Inukurthi. 2024. 3D Scene reconstruction from 2D images. 1, 1 (May 2024), 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

1.1 The Problem

In today's modern science methods, endoscopy has been one of the cornerstones for therapeutic procedures and diagnostics, helping the medical practitioners with visual access to internal organs and tissue information. But, the endoscopy is inherently limited to two-dimensional representation, thereby limiting the depth and spatial understanding of the diagnosis. The transition from 2D to 3D has been a challenge in this field, with researchers trying to bridge the gap to enhance precision of endoscopic procedures. The lack of depth perception has significant challenges in several aspects. Firstly, it complicates the process of localization and characterization of the abnormalities that the tissue may contain thereby making it very difficult to identify and categorize. This limitation leads to difficulties in assessing the tissue morphology, thereby affecting the quality of patient care. By getting access to the three-dimensional reconstruction capabilities to endoscopic imaging, this project seeks to overcome these limitations and get a lot of benefits. With three-dimensional models present, medical practitioners can gain a lot of clear understanding of the anatomical structure by examination, precise treatment planning, etc.

Author's address: Srihari Inukurthi, srihari.inukurthi@ucdenver.edu, University of Colorado Denver, Denver, Colorado, USA.

ACM XXXX-XXXX/2024/5-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1.2 Project Objective

The objective of the project is to create and develop a robust system that can effectively generate the 3-D point clouds so that we can reconstruct the 3D scene from a given group of 2D endoscopic images. This helps in enhanced spatial understanding, improved diagnostic accuracy, and increased quality of patient health care.

1.3 Hypothesis

Lot of existing approaches use supervised learning techniques for building the model for 3D scene reconstruction. In this project, we leverage the use of unsupervised learning techniques for estimating the depth maps and reconstructing 3D scenes using endoscopy images.

1.4 Approach

The project approach starts by first researching and exploring the datasets that are publicly available and are most suitable for its goals. The dataset is then trained using the unsupervised learning methods due to lack of ground truth depth data. The dataset is then converted into a Pytorch dataset, and a data loader is used to read the data. The data is then preprocessed and augmented by using random gamma correction, image flipping, etc. The dataset containing left and right images are then read and transformed into tensors and the images are then used for training. The model training is done on a CUDA GPU and the model that has the best validation loss and the model that has the best loss is saved and used for future predictions. The model that is used employs the use of pretrained ResNet-18md architecture. It uses resnet based encoder and decoder along with learnable upsampling. It utilizes skip connections from encoder by decoder thereby enhancing precise object localization. It uses a loss function that is a combination of SSIM, L-R consistency, disparity smoothness. It also employs a step learning rate scheduler for training. The model is then tested on the samples that are not part of the training and validation dataset. The unseen samples containing 4 channels are then converted to the images that contain 3 channels before testing. The test sample depth maps are then used to generate point clouds that show a 3D representation of the actual endoscopic image implicitly utilizing the global context

2 BACKGROUND

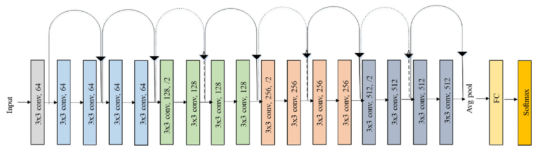
This project uses ResNet18md based encoder and decoders, upsampling them. We also use a combination of loss functions like SSIM, L1-consistency, disparity smoothness, etc. It is beneficial to understand these concepts so that reading the documentation further will be easier.

2.1 ResNet18

ResNet18 stands for Residual Network. It is one of the variants of this family containing 18 layers. The key block of the ResNet18 architecture is the residual block. The main difference between the ResNet and the conventional CNNs is the skip connections which helps in skipping one or more layers so that it is easier to train deeper networks. It is made up of 18 layers containing convolutional layers, pooling layers, fully connected layers, and skip connections. There are 4 residual blocks, where in each block containing 3x3 convolutional layers followed by batch normalization and ReLu activation functions. Also, the skip connections help in vanishing gradient problem by skipping some layers. Generally, the size of the feature maps are big so we reduce the spatial dimensions of the feature maps and at the same time we increase the number of feature maps using downsampling to extract the hierarchical features at different scales effectively. At the end of the network, instead of using the fully connected layers, we use the GAP(Global Average Pooling). This mitigates overfitting while also preserving the spatial information and reducing the number of parameters.

ResNet is pre-trained on ImageNet dataset for extracting the generic features from images. The ResNet18 is very popular among all the variants because of its low cost of computation and the memory requirements making it easier for training on a decent size of dataset. The difference between ResNet18 and ResNet18md is that there is one more layer of lateral shrinkage in the ResNet18md when compared to the original architecture.

Fig. 1. Original ResNet18 Architecture.

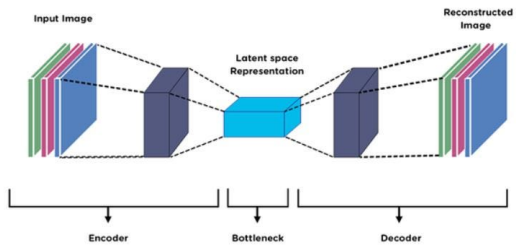


2.2 Encoder-Decoder Architecture

The Encoder-Decoder architecture is a very prominent design in the field of deep learning particularly in tasks like sequence-sequence mapping, image to image translation, semantic segmentation and understanding. The architecture consists of 2 main components: Encoder and Decoder, each dedicated to different aspects of mapping process.

The encoder processes the given input data and encodes into a fixed size representation or a sequence of hidden states. It extracts the high level features, gradually reducing the spatial features while increasing the number of feature maps. And then, the decoder takes the encoded representation and generates the output sequence or output reconstruction. The decoder generates the reconstructed output by upsampling and refining the features to match the desired output.

Fig. 2. Basic Encoder-Decoder Architecture.



2.3 Disparity Smoothness

Disparity smoothness is a common concept widely used in stereo vision and depth estimation tasks. Disparity is referred to as the difference in the horizontal coordinates between corresponding pairs in a pair of stereoscopic images. Disparity maps represent the spatial distribution of disparities across an image pair and have a great impact on downstream tasks like 3-D reconstruction. Disparity maps are filtered using spatial smoothing techniques such as bilateral filtering to cancel noise and enhance local coherence.

3 LITERATURE REVIEW

In general, reconstruction methods traditionally use approaches that are motivated by the image formation from 3-D scene like using the triangulation method and Epi polar geometry. There has been a lot of discussion going on whether machine learning methods are really capable of reconstructing the 3D scene and are actually reliable using the very limited features present in images. Also, most of the approaches use photo-consistency methods to estimate and generate the depth map of the given image with respect to the global context. There has been some great works which include the utilization of the MVG and MVS which are Multi View Geometry and Multi View Stereo respectively. Using these techniques and the technique of ray tracing algorithms, the very first work that has been focused on full 3D reconstruction of the objects is the development of NeRF which abbreviates to Neural Radiance Fields[7]. This uses the technique of overfitting while training the model so that the trained model only contains the weights that are specific to the given set of images that correspond to a single object. The main limitation of using this approach is that ray tracing is expensive and also the model needs to be trained on given set of images for every object. It cannot be used a universal model to estimate for the future predictions/calculations. There have also been some more good works that use the NeRF as the base model and changing the implementation approach for training and the strategy that can be used. Some of the works are Zip-NeRF[2], Mip-NeRF[1] and so on. These NeRF techniques use the Multi Layer Perceptron with back propagation and consider the problem as a supervised regression problem. Also, the notable works that have been done is the 2D3DNet[5]. This primarily focuses on the semantic segmentation of the scene so that only the required information is gathered out of given images. This technique extracts the given image's 2D features by using the segmentation model that has been pre-trained on a similar kind of images and tried to project the 2D features on to the 3D plane and also refines them. SG-NN[3] is also one of the another techniques that uses the TDSF volume is employed that can actually guide the prediction. This also considers the problem as the supervised regression problem. GANs are the most popular approaches that are used these days to solve this problem. GANs allow us to use highly structured adaptable loss functions that showed promising results in the field of scene reconstruction. There are also a lot of commercial and open-source solutions that address these needs. Commercial solutions like Pix4Dmapper, ContextCapture and open source applications like COLMAP, Meshroom provide frameworks for 3D reconstruction using computer vision. Recently, there have been a few approaches that have focused on developing the Deep Learning models that use the unsupervised learning technique instead of the supervised learning approach. One such work is an image synthesis network named DeepStereo[4] that generates the new views by utilizing the pixels from nearby images. While training the model, it tries to predict the camera parameters and use them to predict the nearby image that has not been in the training image. There is also another depth estimation method that named monodepth[6] which uses self supervised learning approach using the encoder-decoder architecture. It uses the depth cues like stereo photometry, geometric constraints to learn the depth from images. It uses a dept consistency loss function enforcing the geometric consistency between predicted depth map and the reconstructed 3D scene. The model is tested on various outdoor benchmark datasets like NYU-Depth-V2 and KITTI.

4 METHOD

4.1 Overview

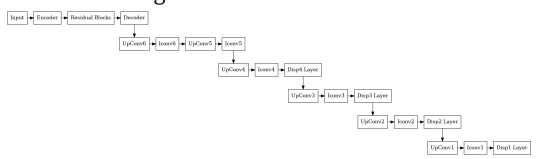
In the field of depth estimation and point cloud generation, there are a lot of existing learning based methods that show promising results, but the only limitation is that these methods treat the problem as a supervised regression problem requiring a lot of depth map(ground truth) data for

training. Since it is always not easy to gather high-quality depth map(ground truth) data under a variety of conditions, in this project we would like to extend the existing approaches by using the more obtainable stereo images instead of the depth data.
The model that is constructed in this project can learn using a sequence of images to generate the depth map without the use of depth map data. The model is trained on image reconstruction loss followed by the use of epi-polar geometry to create disparity maps. And then, the generated disparity maps are used to enhance the accuracy of the model along with the consistency between the generated disparities and the left and right images to improve the performance.

4.2 Architecture

This architecture follows a typical encoder-decoder like architecture, that are augmented with skip connections and the multi scale disparity maps. These are employed to improve the accuracy of the generated depth image. The basic model architecture can be found at 3 The encoder generally starts

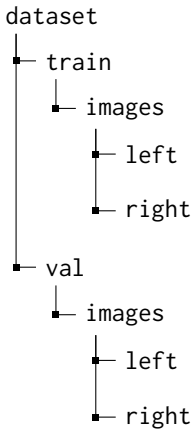
Fig. 3. Model Architecture



with a convolutional layer connected to the batch normalization and activation function. In this case, we have used ELU(Exponential Linear Units) as the activation function. It also includes max pooling layer for downsampling the input. The encoder consists of 4 residual blocks where in every block contains multiple convolutional layers. The decoder uses transpose convolutional layers to upconvolve the encoded feature maps and then concatenates the feature maps(encoded) that are taken from the encoder's skip connections with the upconvolved feature maps. These outputs the intermediate disparity at different stages with different resolutions so that they are processed at the end to generate the depth map.

4.3 Approach

4.3.1 Dataset Exploration, Augmentation and Dataloaders. The dataset that is used to train and evaluate the model consists of a combination of endoscopic images that are captured during minimally invasive surgical procedures like laparoscopy on the gastrointestinal tract and few other internal organs. These images and videos have undergone machine annotations like semantic segmentation and tool tracking prior to the use for 3D reconstruction. the images in the dataset suffer from poor lighting and also a lot of images are not clear enough due to the reflective property of the organs. These videos and images are gathered from open source data repositories like Hamlyn dataset [9] and the Endoslam dataset[8]. Also, there are a few images that are captured from a video sequences that are extracted from the keyframes. The dataset directory structure is as follows:



The dataset contains the left images in left and right images in right. We use a total of 17923 images for training and 4635 images for validation. The dataset is used to create a PyTorch dataset by extracting the images from left and right directory and are transformed to create a dataloader that will be used to train the network. When the images are extracted, all the images undergo different stages like preprocessing, augmentation, concatenation, etc. All the images are resized to the size (256, 512) and augmented by using the techniques like flipping the image horizontally and vertically with a random degree and also augmenting the image pairs using augment parameters. These images are converted to tensors that are used for training. The code snapshots are found at 4 and 5.

Fig. 4. Dataset augmentation.

```
def create_image_transforms(mode='train', augment_params=[0.8, 1.2, 0.5, 2.0, 0.8, 1.2],
                            do_augmentation=True, size=(256, 512)):
    if mode == 'train':
        transform = transforms.Compose([
            ResizeImage(train=True, size=size),
            RandomFlip(do_augmentation),
            ToTensor(train=True),
            AugmentImagePair(augment_params, do_augmentation)
        ])
    elif mode == 'test':
        transform = transforms.Compose([
            ResizeImage(train=False, size=size),
            ToTensor(train=False),
            DoTest(),
        ])
    return transform
```

Fig. 5. Dataloader generation.

```
def prepare_dataloader(data_directory, mode, augment_params, do_augmentation, batch_size, size, num_workers):
    data_dirs = os.listdir(data_directory)
    transform = create_image_transforms(
        mode=mode,
        augment_params=augment_params,
        do_augmentation=do_augmentation,
        size=size)
    datasets = [DepthDataset(os.path.join(data_directory, data_dir), mode, transform=transform)
                for data_dir in data_dirs]
    dataset = ConcatDataset(datasets)
    n_img = len(dataset)
    print('Use a dataset with', n_img, 'images')
    loader = DataLoader(dataset, batch_size=batch_size,
                        shuffle=(mode == 'train'), num_workers=num_workers,
                        pin_memory=True)
    return n_img, loader
```

4.4 Model

The main idea behind the model construction is the reconstruction based architecture from the left images where a left-right disparity map is generated and using the left image and the disparity map, we try to reconstruct the right image. The estimated right image is then compared against

the right image that is already present and then the loss is calculated to fine tune the model. We used the left-right disparity consistency and using a bilinear sampler to generate the images that can be differentiable. The model uses ResNet-based architecture combined with convolutional and upconvolutional layers. The convolutional layers are generally followed by batch normalization layers and the ELU activation functions. The encoder utilizes layers from pretrained ResNet model to extract features at different scales. The decoder has upconvolutional layers to generate disparity maps at multiple scales. Skip connections are used to fuse features that are taken from the encoder with corresponding layers in the decoder. The model predicts disparity maps at multiple scales. The disparity maps that are representing the differences in the pixel coordinates horizontally between corresponding points in the images are then use to estimate the depth maps. The disparity maps are constructed at the end of each block. The 4 blocks generate 4 disparity maps which are then processed to create a single disparity map that is then after used for estimating the depth map image. The disparity maps are then used to compute the average disparity between the given right and left images so that we can get an disparity map that is smooth. Then, we use the weights to the pixels that are closer to the edges so that we can see a clear difference in the depth map. These techniques are employed so that we can fill out any missing depth information for some pixels without compromising on the information obtained from the generated left and right disparity maps. We use different kind of loss functions and assign weights to each of the 3 different loss functions. We generate the image pyramid with four scales using the bilinear interpolation and compute the horizontal and vertical gradients for the image. Using these gradients, we try to compute the smoothness of the disparity maps that are generated by warping the images. The model code snapshot can be found at 6.

Fig. 6. The Deep learning model.

```
# encoder
self.encoder1 = resnet.layer1 # H/4
self.encoder2 = resnet.layer2 # H/8
self.encoder3 = resnet.layer3 # H/16
self.encoder4 = resnet.layer4 # H/32

# decoder
self.upconv6 = upconv(filters[3], 512, 3, 2)
self.iconv6 = convolution(filters[2] + 512, 512, 3, 1)

self.upconv5 = upconv(512, 256, 3, 2)
self.iconv5 = convolution(filters[1] + 256, 256, 3, 1)

self.upconv4 = upconv(256, 128, 3, 2)
self.iconv4 = convolution(filters[0] + 128, 128, 3, 1)
self.disp4_layer = get_disparity(128)

self.upconv3 = upconv(128, 64, 3, 1) #
self.iconv3 = convolution(64 + 64 + 2, 64, 3, 1)
self.disp3_layer = get_disparity(64)

self.upconv2 = upconv(64, 32, 3, 2)
self.iconv2 = convolution(64 + 32 + 2, 32, 3, 1)
self.disp2_layer = get_disparity(32)

self.upconv1 = upconv(32, 16, 3, 2)
self.iconv1 = convolution(16 + 2, 16, 3, 1)
self.disp1_layer = get_disparity(16)
```

4.5 Loss functions

We use the left and right images while training to compute the loss. The right image is used to compute the loss by predicting the l-r disparities using only left image and employing a bilinear sampler.

4.5.1 SSIM(Structure Similarity Index Measure. Structure Similarity Index is a very widely used metric in a lot of computer vision algorithms to perceive the quality of images. It checks the similarity between 2 images based on their structural information. The SSIM is calculated based on the structure comparison for the reference and distorted images. Not only the structure comparison, it also uses the contrast and the luminance of the images. The values range from -1 to 1 where the most similar image has a value closer to 1 and the most dissimilar image has a value closer to -1. The mathematical calculation is done using the following formula 1

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + d_1)(2\sigma_{xy} + d_2)}{(\mu_x^2 + \mu_y^2 + d_1)(\sigma_x^2 + \sigma_y^2 + d_2)}, \tag{1}$$

where x and y are the images that are to be compared, σ_x^2 and σ_y^2 are the variances of x and y respectively, μ_x and μ_y are the mean values of x and y respectively, σ_{xy} is the covariance of x and y, d_1 and d_2 are the little constants that are applied to the denominator so that we can prevent instability if in case the denominator tends to approach zero.

4.5.2 Disparity smoothness. Disparity smoothness is mainly used in stereo vision and depth estimation algorithms. It calculates the difference in the horizontal coordinates between the corresponding points in a pair of stereoscopic images that are due to relative displacement of objects between two viewpoints. The assumption with respect to the disparity maps is that the neighboring pixels in the map should have similar disparities if they belong to the same object or surface in the scene. We use the bilateral filtering for spatial smoothing to enhance local coherence and suppress noise. The filtering operations obscure the difference values whereas protecting the edges and discontinuities so that we can generate smoother and visually appealing depth maps.

4.5.3 L-R disparity consistency. The Left-Right disparity consistency is a well-known concept in stereo vision and depth estimation algorithms. It is referred to as the constraint that the disparity maps that are computed using the left and right images are consistent with each other. It helps in identifying and correcting errors or inconsistencies in disparity maps to enhance spatial coherence. The mathematical formula is done using the equation 2.

$$D_L(x, y) = -D_R(x - D_L(x, y), y) \tag{2}$$

where $D_L(x, y)$ is the disparity of a point(x,y) in left image and $D_R(x, y)$ is the disparity of the same point(x,y) in the right image. This equation refers that the disparity value of a point that is considered in the left image should be equal to the negative of the disparity of the same point in the right image shifted by the left image disparity value. The loss functions snapshot can be found at 7.

4.6 Implementation

The implementation is done using a conventional encoder decoder architecture where the encoding phase consists of layers from the pretrained ResNet18 model and the decoder consists of upconvolutional and convolutional layers. The model consists of convolution blocks containing convolutional layers followed by batch normalization and activation. These blocks are applied with zero padding to ensure that the spatial dimensions of input and output match. The upconv block implements an upsampling convolutional layer followed by a convolution operation. We use 3 input channels to

Fig. 7. The Loss functions.

```
def forward(self, input, target):
    left, right = target
    left_pyramid = self.scale_pyramid(left, self.n)
    right_pyramid = self.scale_pyramid(right, self.n)
    disp_left_est = [d[:, 0, :, :].unsqueeze(1) for d in input]
    disp_right_est = [d[:, 1, :, :].unsqueeze(1) for d in input]
    left_est = [self.generate_image_left(right_pyramid[i], disp_left_est[i]) for i in range(self.n)]
    right_est = [self.generate_image_right(left_pyramid[i], disp_right_est[i]) for i in range(self.n)]
    right_left_disp = [self.generate_image_left(disp_right_est[i], disp_left_est[i]) for i in range(self.n)]
    left_right_disp = [self.generate_image_right(disp_left_est[i], disp_right_est[i]) for i in range(self.n)]
    disp_left_smoothness = self.disp_smoothness(disp_left_est, left_pyramid)
    disp_right_smoothness = self.disp_smoothness(disp_right_est, right_pyramid)
    l1_left = [torch.mean(torch.abs(left_est[i] - left_pyramid[i])) for i in range(self.n)]
    l1_right = [torch.mean(torch.abs(right_est[i] - right_pyramid[i])) for i in range(self.n)]
    ssim_left = [torch.mean(self.SSIM(left_est[i], left_pyramid[i])) for i in range(self.n)]
    ssim_right = [torch.mean(self.SSIM(right_est[i], right_pyramid[i])) for i in range(self.n)]
    image_loss_left = [self.SSIM_w * ssim_left[i] + (1 - self.SSIM_w) * l1_left[i] for i in range(self.n)]
    image_loss_right = [self.SSIM_w * ssim_right[i] + (1 - self.SSIM_w) * l1_right[i] for i in range(self.n)]
    image_loss = sum(image_loss_left + image_loss_right)
    lr_left_loss = [torch.mean(torch.abs(right_left_disp[i] - disp_left_est[i])) for i in range(self.n)]
    lr_right_loss = [torch.mean(torch.abs(left_right_disp[i] - disp_right_est[i])) for i in range(self.n)]
    lr_loss = sum(lr_left_loss + lr_right_loss)
    disp_left_loss = [torch.mean(torch.abs(disp_left_smoothness[i])) / 2 ** i for i in range(self.n)]
    disp_right_loss = [torch.mean(torch.abs(disp_right_smoothness[i])) / 2 ** i for i in range(self.n)]
    disp_gradient_loss = sum(disp_left_loss + disp_right_loss)
    loss = image_loss + self.disp_gradient_w * disp_gradient_loss + self.lr_w * lr_loss
    return loss
```

initialize the model and a pre-trained resnet model layers in the encoder. The encoder comprises of 4 layers of the ResNet model and downsample the spatial dimensions of the feature maps by a factor of 4, 8, 16, 32 respectively. The decoder comprises of a series of upconvolutional and convolutional layers as blocks with filters of 512 and 256 with alternating strides of 2 and 1 respectively. The decoder also consists of a disparity layer that computes the disparity map of the particular block such that we get four disparity maps that come at the end of each epoch. We also use max pooling and skip connections in the encoder. All the layers use ELU or ReLU as the activation functions. We perform forward and backward passes for every epoch and compute the loss function by using the combination of the loss functions at 1.

The network is trained using Adam Optimizer starting with an initial learning rate of 0.01 and is adjusted periodically. We use exponential decay to dynamically adjust the learning rate based on the epochs. The learning rate is reduced by a factor of 2 depending on the epoch count. This helps in the model learning better and yielding better results rather than that of having a fixed learning rate. The disparity map that is generated is refined for better visualization and accuracy. The disparity maps are enhanced by applying specific masks and blending techniques to enhance the depth prediction by generating the refined version of the depth map. The model containing the best validation loss is stored for future computations. GPU acceleration was employed during the training phase to expedite the computational workload since there were a lot of images for the model to be trained on and also the model was not very small. We have used one GPU with CUDA version 12.4.

We use open3d geometry for generating the point clouds that are computed using the RGB image(reference image) and the depth map(generated). To generate the point cloud, we first try to create a RGB-D image that is calculated using the RGB image(reference) and the corresponding depth map image. The RGB-D image is then used to generate the 3D structure using the formulae 3.

$$
\begin{aligned}
z &= d/depthscale \\
x &= (u - c_x) * z/f_x \\
y &= (v - c_y) * z/f_y
\end{aligned} \tag{3}
$$

where (x,y,z) represents the point in a 3D coordinate system and d represents a depth point corresponding to the point image point (u, v). f_x, c_x, f_y and c_y are the camera parameters.

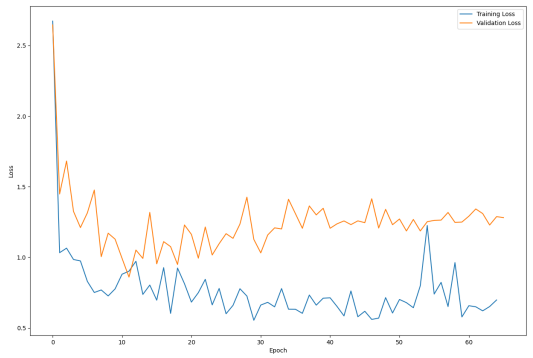
5 RESULTS AND DISCUSSIONS

At first, we have employed different versions of the resnet baseline model like resnet50, resnet34, etc to test which suits better for the task. Since, we are using layers from the pre-trained model

to use in the encoder, we felt that it would be better to have resnet18 because it is a small model and would be easier to train. We have used spatial attention mechanism instead of the channel attention because it aligns better with the task. For the purpose of evaluation, we train and test our method on different endoscopic images that are extracted from Hamlyn data[9], EndoSlam data[8] and also few images only for testing from a private dataset EndoVis. The model is trained on 17923 endoscopic images. Also, we considered the left and right images of stereo sequences as independent images. While testing the model on images that are from the real time, we transformed the images from four channels to 3 channels because the model is trained to consider input images that contain only 3 input channels. We use only left images for estimating the depth map. While producing the depth map, the images are squeezed to contain the dimensions of the original RGB image so that it is easier to generate point clouds using the RGB image and the corresponding depth map. The point cloud generation step encounters a problem of incompatibility if the RGB image and the depth map does not contain same dimensions.

The model's evaluation generated results that showed a satisfactory performance and precision in estimating the depth maps and generating the point clouds for provided images(either left or right). The plot that shows the change in loss while training is shown in 8.

Fig. 8. Loss Plot



The model has been evaluated on a handful of samples that are picked from the EndoSlam dataset[8] and the Hamlyn dataset[9] that are not previously seen by the model while training. The samples that are picked also contain the ground truth depth maps so that it is easier to compare the generated results with the ground truth depth map data and compute the evaluation metrics.

5.1 Evaluation metrics

The following metrics have been adopted to evaluate the quality of the estimated depth maps and compute the model's capability of generating depth maps that are closer to the ground truth.

5.1.1 Root Mean Square Error(RMSE). This is a very popular evaluation metric that is used in most of the regression type problems. The metric refers to the standard deviation of the difference between the generated/estimated depth map to that of the ground truth depth map image.

$$\mathrm{RMSE}(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{N}} \tag{4}$$

Here, y represents the ground truth depth map image and $\hat{y}$ represents the estimated depth map image. y_i represents the ground truth depth value at i-th pixel in the image and $\hat{y}_i$ represents the

estimated depth value at i-th pixel from the image. N represents the minimum of the number of grid cells that are present in both images. This is because if the number of grid cells in both the images are different, then the error might be very huge.

The RMSE of the given sample is 8.381. Since we are estimating the depth maps from an approach that is based on unsupervised learning, the RMSE is pretty decent. The methods that use supervised learning approach have the average RMSE values that are in the range of 5-8. The evaluation of the model on a sample set of 20 images that are not included in the training showed the RMSE value of 10.279.

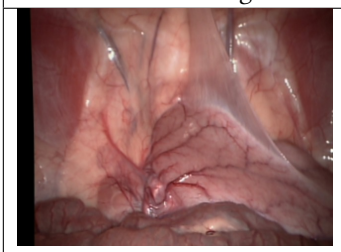
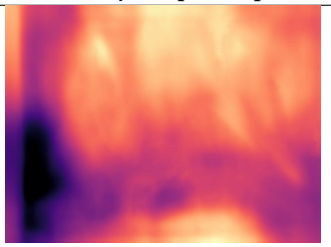
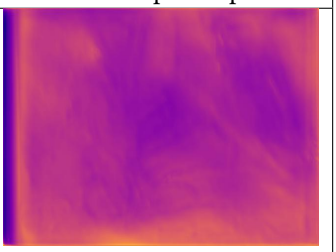
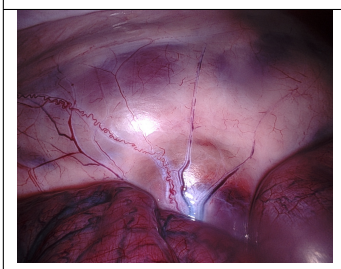
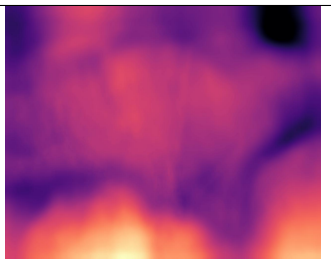
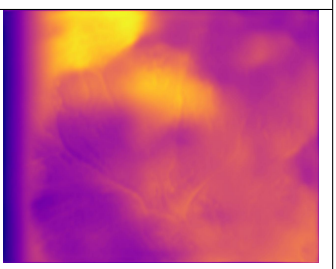
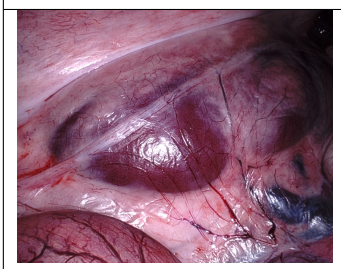
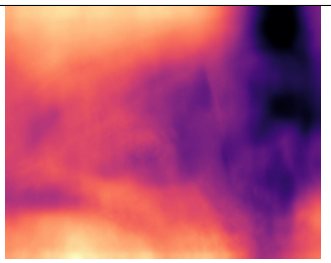
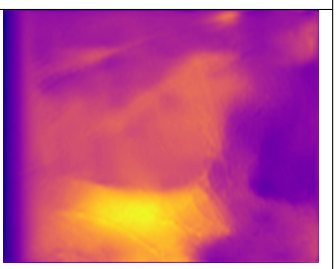
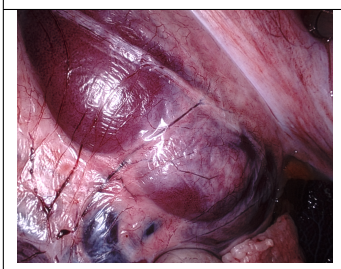
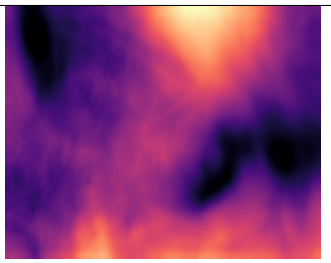
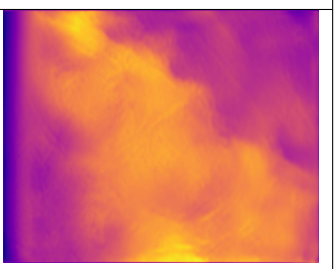
Reference image	Hamlyn depth map	Our depth map
		
		
		
		

Table 1. Results Comparison

Table 1 shows the qualitative results of our depth estimation algorithm versus the other alternatives. Specifically, we compare it against the Hamlyn implementation[9]. The results show that the model was able to generate depth maps that are decent enough to compete with the other approach. The point clouds are generated using the open3d geometry by using the RGB image(reference) and the depth map along with the camera's calibration parameters. The point cloud that is generated from the given test sample above is at 9 Although, there was no ground truth for the scene and

Fig. 9. Point cloud.

accurate camera trajectory, the accuracy of the reconstruction can be qualitatively assessed in the figure 9. We can see that there is a higher accuracy for reconstruction than the other approaches that are trained on the outdoor sequences which highlights the potential of the depth networks and the volumetric tracking for the point cloud generation and motion estimation from monocular endoscopies. The Python implementation for the training took 14 minutes for each epoch and the training is done for 65 epochs. The point cloud generation took 500 milli seconds in our GPU(Nvidia GeForce GTX 1080) for 1310720 points. We couldn't evaluate the generated point clouds since there was no ground truth. Also, to compute the closeness of the point clouds, we need to evaluate the accuracy of the hundreds and millions of 3D points which is computationally very expensive.

6 CONCLUSIONS

6.1 Summary

The project mainly focuses on image-based 3D scene reconstruction, specifically focused developing a framework on endoscopic image 3D scene reconstruction using the deep learning methods. This project tries to address the limitations of using existing supervised learning approaches which basically rely on ground truth depth maps and the use of conventional stereo or multi-view stereo matching algorithms. Utilizing diverse samples from different parts of body using the endoscopic images, the model extracts the spatial feature maps and reduce them using the encoder from the pretrained ResNet model. The model shows good results when compared to the existing approaches that use supervised learning methods.

6.2 Potential Impact

This project can impact and change the existing medical facilities and system a lot by providing various benefits across various fields of medical facilities. Some of the benefits include:

6.2.1 Improved diagnostic accuracy. 3D scene allows us to make more comprehensive visualization and examination of the therapeutic procedures helping in increasing the accuracy of the diagnostics. Clinicians can get deeper understanding of complex anatomical structures, identify the variations in the body that might be different from the original norm and make more precise diagnosis, especially in cases where it is hard to visualize what is inside the body.

6.2.2 Surgical navigation and simulation. Surgeons can actually use these models to plan and simulate complex surgical procedures that help in improving the preoperative planning and guidance for intraoperative planning. Also, real-time navigation based on these 3D reconstructed models help in enhancing surgical precision, reducing the operative time, minimizing the tissue damage and also increase the success rate of surgery a lot.

7 LIMITATIONS

The main limitation is the availability and the quality of the data that is used for training. The resolution of the images is not very high when compared to the datasets that are generally used for the 3D scene reconstruction. Also, the lighting conditions, background objects affect the performance of training. The other limitation is that the model tends to perform well on a specific type of images that do not exhibit any kind of reflecting property in them. Additionally, the computational complexity of the training is very high and it requires a lot of compute resources for training and inference.

8 FUTURE RESEARCH

The proposed approach provides as a baseline model for delving into the unsupervised learning approach in the field of 3D scene reconstruction from endoscopic images. The further step in this direction would be to perform semantic understanding of the 3D scene that is generated. This approach only provides us the information of the scene based off of an image, but in real-time, it is required to be able to reconstruct the scene from a video sequence by being able to estimate the camera calibration parameters and generate point clouds. The current model might not work well with images that contain occlusions, the next step would be able to handle the occlusions that are present in the provided scene.

REFERENCES

[1] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2022. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. https://doi.org/10.48550/arXiv.2111.12077 arXiv:2111.12077 [cs].

[2] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. 2023. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. https://doi.org/10.48550/arXiv.2304.06706 arXiv:2304.06706 [cs].

[3] Angela Dai, Christian Diller, and Matthias Nießner. 2020. SG-NN: Sparse Generative Neural Networks for Self-Supervised Scene Completion of RGB-D Scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE.*

[4] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. 2015. DeepStereo: Learning to Predict New Views from the World's Imagery. https://doi.org/10.48550/arXiv.1506.06825 arXiv:1506.06825 [cs].

[5] Kyle Genova, Xiaoqi Yin, Abhijit Kundu, Caroline Pantofaru, Forrester Cole, Avneesh Sud, Brian Brewington, Brian Shucker, and Thomas Funkhouser. 2021. Learning 3D Semantic Segmentation with only 2D Image Supervision. https://doi.org/10.48550/arXiv.2110.11325 arXiv:2110.11325 [cs].

[6] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. 2017. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *CVPR.*

[7] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. https://doi.org/10.48550/arXiv.2003.08934 arXiv:2003.08934 [cs].

[8] Kutsev Bengisu Ozyoruk, Guliz Irem Gokceler, Taylor L. Bobrow, Gulfize Coskun, Kagan Incetan, Yasin Almalioglu, Faisal Mahmood, Eva Curto, Luis Perdigoto, Marina Oliveira, Hasan Sahin, Helder Araujo, Henrique Alexandrino, Nicholas J. Durr, Hunter B. Gilbert, and Mehmet Turan. 2021. EndoSLAM dataset and an unsupervised monocular visual odometry and depth estimation approach for endoscopic videos. *Medical Image Analysis* 71 (July 2021), 102058. https://doi.org/10.1016/j.media.2021.102058

[9] David Recasens, José Lamarca, José M. Fácil, J. M. M. Montiel, and Javier Civera. 2021. Endo-Depth-and-Motion: Reconstruction and Tracking in Endoscopic Videos using Depth Networks and Photometric Constraints. https://doi.org/10.48550/arXiv.2103.16525 arXiv:2103.16525 [cs].

Received 08 May 2024