3D Color Point Cloud Prediction and Registration

Avinash Ayite
Masters in Robotics
Northeastern University

Tarun Srinivasan
Masters in Robotics
Northeastern University

Jose Joseph Thandapral
Masters in Robotics
Northeastern University

1 Introduction

As modern-day autonomous systems move to camera based solutions due to their ease of use and rich information. However, accurate depth estimation from monocular and stereo videos is needed for more complex tasks like mapping, control and registration. The goal of our project is to address two challenges in the field of autonomous systems that rely on camera-based solutions: accurate depth estimation from monocular and stereo videos, and registration of obtained point clouds to map the environment.

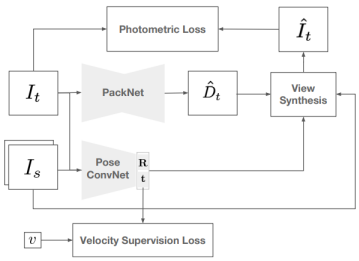
To achieve this, we plan to implement various deep learning networks in our project. For monocular depth estimation, we have selected two state-of-the-art networks, PackNet [2] and MiDaS[5], which have shown promising results in estimating accurate depth maps from single monocular images. Additionally, we also plan to implement a stereo depth estimation network, RAFT-Stereo[3], which utilizes stereo image pairs to estimate depth maps.

Once we obtain depth maps from monocular or stereo networks, the next step in our project is to perform point cloud reprojection and registration. Point cloud reprojection involves projecting the obtained depth maps onto a 3D point cloud representation, which allows us to obtain a dense 3D representation of the environment. Point cloud registration is the process of aligning these obtained 3D point clouds to a pre-built map or reference frame, allowing us to accurately localize the camera in the environment and perform mapping and control tasks.

2 Models

2.1 PackNet [2]

PackNet is scale-aware selfsupervised monocular structure-from-motion model, which predicts depth and camera pose, and optionally includes weak velocity supervision at training time to produce scale-aware depth and pose. It has 128,294,020 parameters for depth prediction and 1,592,684 parameters for Pose estimation.



	Layer Description	K	Output Tensor Dim.
#0	Input RGB image		3×H×W
	Encoding Layers		
#1	Conv2d	5	64×H×W
#2	Conv2d → Packing	7	64×H/2×W/2
#3	ResidualBlock (x2) → Packing	3	64×H/4×W/4
#4	ResidualBlock (x2) → Packing	3	128×H/8×W/8
#5	ResidualBlock (x3) → Packing	3	256×H/16×W/16
#6	ResidualBlock (x3) → Packing	3	512×H/32×W/32
	Decoding Layers		
#7	Unpacking (#6) → Conv2d (⊕ #5)	3	512×H/16×W/16
#8	Unpacking (#7) → Conv2d (⊕ #4)	3	256×H/8×W/8
#9	InvDepth (#8)	3	1×H/8×W/8
#10	Unpacking (#8) → Conv2d (⊕ #3 ⊕ Upsample(#9))	3	128×H/4×W/4
#11	InvDepth (#10)	3	1×H/4×W/4
#12	Unpacking (#10) → Conv2d (⊕ #2 ⊕ Upsample(#11))	3	64×H/2×W/2
#13	InvDepth (#12)	3	1×H/2×W/2
#14	Unpacking (#12) → Conv2d (⊕ #1 ⊕ Upsample(#13))	3	64×H×W
#15	InvDepth (#14)	3	1×H×W

Figure 1: PackNet Architecture [2]

PackNet uses a novel way to downsample and upsample, detail-preserving 3D convolution layers using packing and unpacking modules. PoseNet is a proposed convolutional neural network which was trained to regress ego-motion for any two context frames. It is a 7-layer convolutional network, ending with a 1x1 convolution layer with a regression pose output between each from context and target frame.

View synthesis reconstructs the image in the target view, by sampling pixels from I_s context frame. $\hat{I}_t$ is calculated based on the predicted depth map $\hat{D}_t$ and the relative pose $T_{t \to s}$. This reconstructed image is used to calculate photometric loss which is then used for backpropogation to train the model. The objective function is a combination of different loss terms, masks and regularization terms which defined in PackNet [2].

2.2 Dense Vision Transformers (MiDaS) [5]

Dense vision transformers, an architecture that leverages vision transformers in place of convolutional networks as a backbone for dense prediction tasks. Which assembles tokens from various stages of the vision transformer into image-like representations at various resolutions and progressively combine them into full-resolution predictions using a convolutional decoder. The transformer backbone processes representations at a constant and relatively high resolution and has a global receptive field at every stage. These properties allow the dense vision transformer to provide finer-grained and more globally coherent predictions when compared to fully-convolutional networks

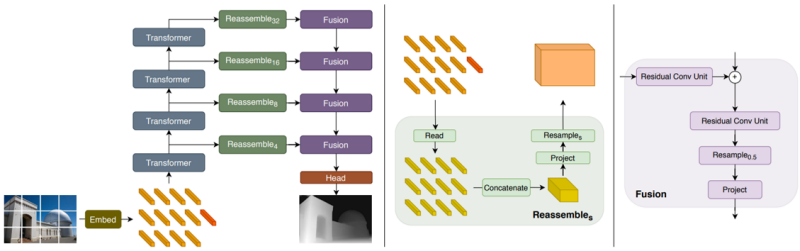


Figure 2: DPT Architecture [5]

The input image is transformed into tokens (orange) either by extracting non-overlapping patches followed by a linear projection of their flattened representation (DPT-Base and DPT-Large) or by applying a ResNet-50 feature extractor (DPT-Hybrid). The image embedding is augmented with a positional embedding and a patch-independent readout token (red) is added. The tokens are passed through multiple transformer stages. The reassemble tokens from different stages into an image-like representation at multiple resolutions (green). Fusion modules (purple) progressively fuse and upsample the representations to generate a fine-grained prediction. Overview of the Reassembles operation. Tokens are assembled into feature maps with $1/s$ the spatial resolution of the input image. Fusion blocks combine features using residual convolutional units and upsample the feature maps

2.3 RAFT Stereo [3]

A new deep architecture for rectified stereo based on the optical flow network RAFT. Uses a multi-level convolutional GRUs, which more efficiently propagate information across the image. Given a pair of rectified images (I_L, I_R), estimate a disparity field d giving the horizontal displacement for every pixel in I_L. Similar to RAFT this approach is composed of three main components: a feature extractor, a correlation pyramid, and a GRU-based update operator as shown in Fig. 3. The update operator iteratively retrieves features from the correlation pyramid and performs updates on the disparity field.

Two separate feature extractors termed the feature encoder and the context encoder are used. The feature encoder is applied to both the left and right images and maps each image to a dense feature map, which is then used to construct the correlation volume.

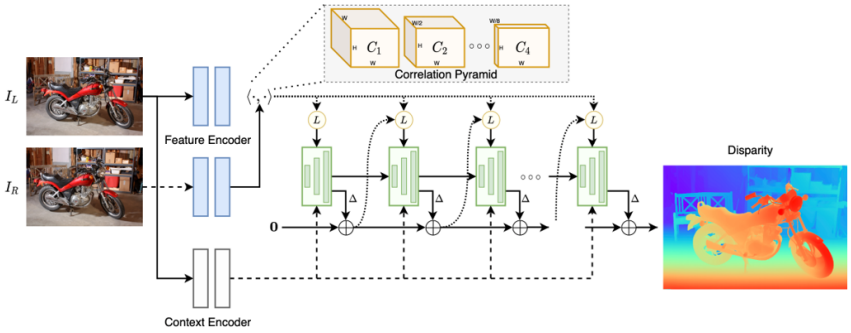


Figure 3: RAFT-Stereo Architecture [3]

The context encoder has identical architecture to the feature encoder except we replace instance normalization with batch normalization and only apply the context encoder on the left image. The context features are used to initialize the hidden state of the update operator and also injected into the GRU during each iteration of the update operator. Supervision is provided as L_1 distance between the predicted and ground truth disparity over the full sequence of predictions with exponentially increasing weights.

3 Approach

3.1 Point Cloud Reprojection

Form the depth images, we reproject image points to 3D using the calibration parameters. The RGB colour information was extracted from the corresponding image to get the pointcloud. This script was written in Python using libraries like Numpy and Open3D.

The equations we used to convert image points (x,y) to 3D (X,Y,Z) is as follows:

$$Z = \text{pixel value form the depth map}$$
$$X = Z(x - cx)/fx$$
$$Y = Z(y - cy)/fy$$

Where *fx, fy, cx, cy* are all intrinsic camera parameters.

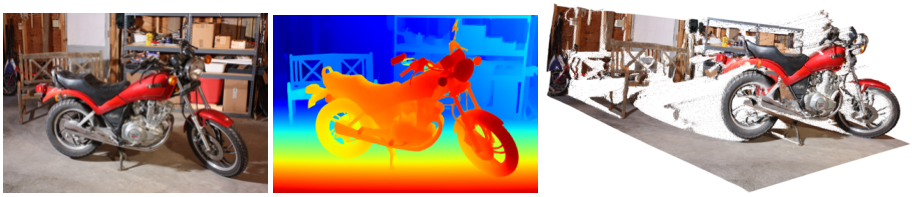


Figure 4: Result from our Reprojection Script and corresponding RGB Image and Death map

3.2 Point Cloud Registration

3.2.1 Global Registration

Global registration methods do not rely on an initial alignment for initialization. Instead, they consider the overall structure and geometry of the entire point clouds. These methods aim to estimate a transformation that aligns the entire point clouds, taking into account the overall shape, orientation, and scale of the point clouds, rather than relying on local point-to-point correspondences.

For global registration we use RANSAC algorithm. In each RANSAC iteration, random points are picked from the source point cloud. Their corresponding points in the target point cloud are detected by querying the nearest neighbor in the 33-dimensional FPFH feature space. A pruning step takes fast pruning algorithms to quickly reject false matches early.

3.2.2 Colored ICP Registration [4]

ICP (Iterative Closest Point) registration and Colored point cloud registration are local registration methods that require an initial rough alignment as initialization. The color information locks the alignment along the tangent plane. Thus this algorithm is more accurate and more robust than prior point cloud registration algorithms, while the running speed is comparable to that of ICP registration.

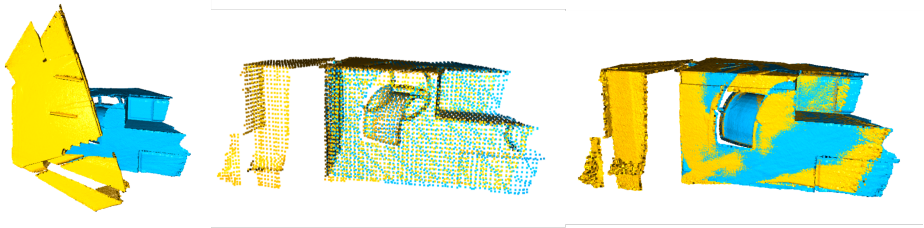


Figure 5: From Left to Right: Initialize with global registration and refine with colored ICP

4 Dataset

The KITTI [1] benchmark is the de facto standard for depth evaluation. This has 39810 images for training, 4424 for validation and 697 for evaluation. We evaluate PackNet [2], MiDaS [5] and RAFT-Stereo [3] using KITTI benchmark dataset [1] which includes ground truth depth maps and camera poses. In order to evaluate the generalization of model we propose to use a real time video stream from a Logitech camera which is calibrated using traditional checkerboard calibration.

4.1 Camera Calibration - Logitech

Camera calibration is the process of determining the relationship between the 3D world and the 2D image captured by a camera. Using a checkerboard pattern is a popular method of calibration, as it provides a precise and easy-to-detect set of features.

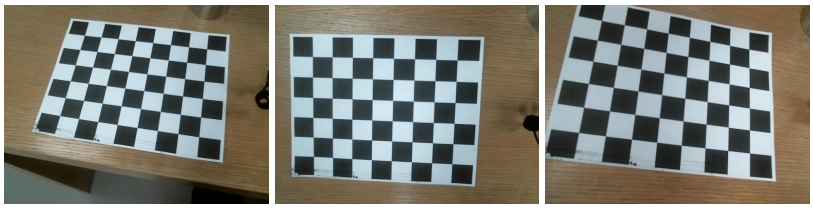


Figure 6: Input images before calibration - 640x480

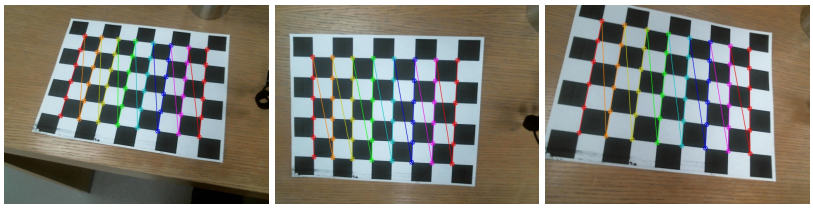


Figure 7: Calibrating the Logitech WebCam - Corner detection

The process involves taking images of the checkerboard from different angles and then analyzing the distortions caused by the camera's optics and position. By measuring these distortions, the intrinsic and extrinsic parameters of the camera can be calculated, allowing for more accurate measurements and 3D reconstruction.

```
Camera matrix :

[[658.94546219   0.         325.66435424]
 [  0.         661.46732583 229.53852534]
 [  0.           0.           1.        ]]
dist :

[[-6.60778320e-03 -8.55788894e-01 -9.06606778e-04  2.36146657e-04
   3.43531640e+00]]
total error: 0.02830681789820365
(o3d) avinash@pc:~/Desktop/calibration$
```

Figure 8: Result from Camera calibration

We were able to achieve a re-projection error of 0.02 pixels, which shows that the camera is calibrated accurately enough to use in PackNet for Depth prediction.

5 Evaluation Metrics

For this application, we choose from popular metrics used in the field to evaluate the output Depth Maps to provide a quantitative idea of how the result is compared to a given ground truth depth map. We also compare Frame rates for the different models and the resolutions they support during inference.

- RMSE: Root mean squared error is a well known metric in the Machine Learning field, which is nothing but the L2 norm between GT and predicted depth values.
- a_1 or δ_1: This is defined as the percentage of predicted depth values whose relative error ratio (i.e., the max of the ratio between the predicted and ground truth depth values) is smaller than a threshold of 1.25.

$$\text{Threshold: } \% \text{ of } y_i \text{ s.t. } \max(\frac{y_i}{y_i^*}, \frac{y_i^*}{y_i}) = \delta < thr$$

$$\text{RMSE (linear): } \sqrt{\frac{1}{|T|}\sum_{y \in T} ||y_i - y_i^*||^2}$$

Figure 9: Evaluation metrics chosen

6 Experiments

6.1 Environment Setup

The source code that we have used for this project is available. The compute used for this project was provided by the Discovery Cluster at Northeastern on a V100-SMX2 with 32GB of memory and our personal Laptop setup, with a Nvidia RTX 3050Ti GPU with 4GB GPU memory. As Docker was not supported by the cluster here, we created an Anaconda environment and a singularity container to run this implementation.

For real time inference, we set up a web socket using Droid CAM to send and receive video feed from an iPhone 14 Pro which was then sent to the networks for real-time depth prediction.

6.2 PackNet Inference

As the initial experiment, we proceed to evaluate the PackNet implementation on the KITTI dataset and then test it on our Logitech camera that we had calibrated. The self-supervised network was pre-trained using the training set in KITTI and weights released along with the source code. This was downloaded and imported onto the default model and was used to evaluate on the test set. The images in KITTI were resized to a smaller dimension, i.e, 192x640 for all training and evaluation purposes. The output, which were depth images for all the input files, were saved and visualized. The single channel depth map was of the same size as the input with each pixel denoting depth at that point in meters.

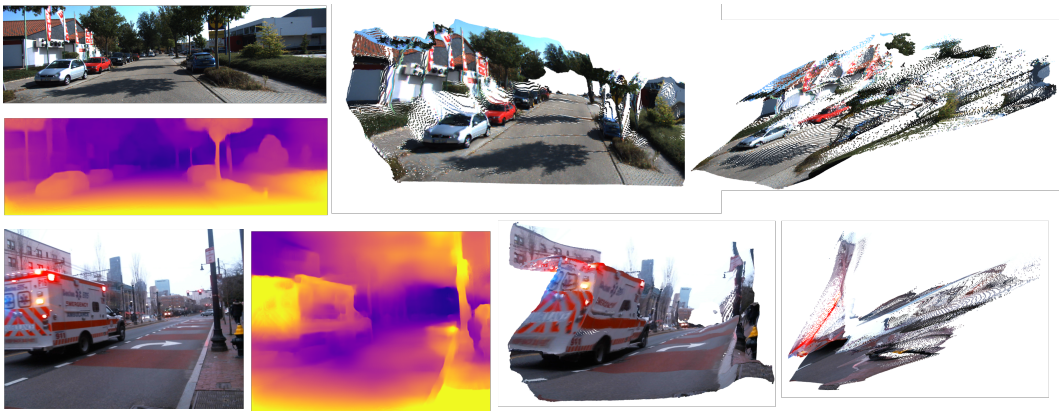


Figure 10: Depth Maps and Reprojected Point Clouds from KITTI and LogiTech Camera.

6.3 MiDaS Inference

The depth maps recieved in PackNet were not very accurate for point cloud reprojection and moreover were running at 4 frames per second. We chose MiDaS as the authors had claimed that the models were generalized across different environments and also could run in real-time. We set up an environment using DROID-CAM which connects an iPhone camera and our local PC to run inference in real-time. The input images were full scale, at 640x480 resolution and the output depth maps had the same resolution. But the important thing we noticed here was that MiDaS could generalize between different contexts and different resolutions across datasets.

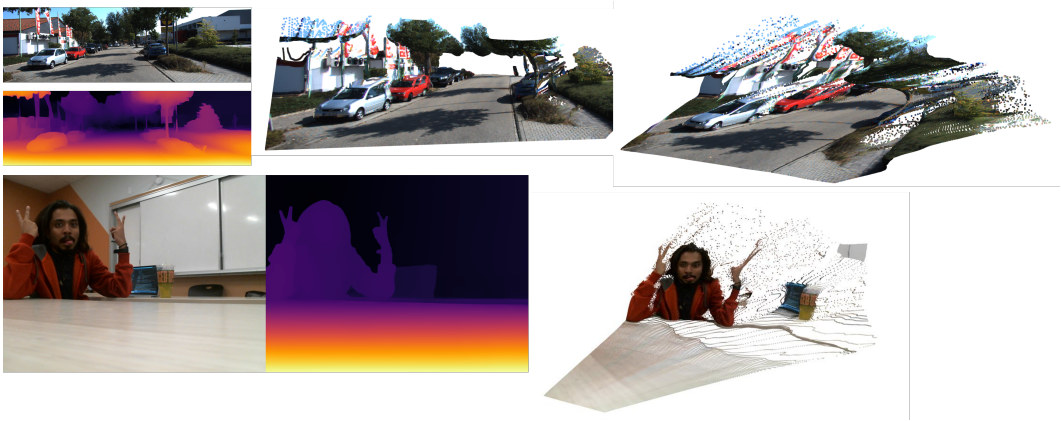


Figure 11: Depth Maps and Reprojected Point Clouds from KITTI and LogiTech Camera. Video link

6.4 RAFT-Stereo Inference

As we found that both of the monocular networks we had tested were able to produce viable point clouds for registration, we chose to move to stereo options, out of which RAFT-Stereo stood out as a promising option. RAFT uses optical flow between stereo pairs to predict disparity maps. This fact helped RAFT produce much better and accurate disparity maps, which in turn gave better point clouds which we could use further down the pipeline. The input to this model would be stereo pairs of images in full scale, at 1242x375 for the KITTI dataset. Inference time clocked at about 2 frames per second to get the disparity maps. We also note that there was a trade-off between frame rate and accuracy, to get better point clouds.

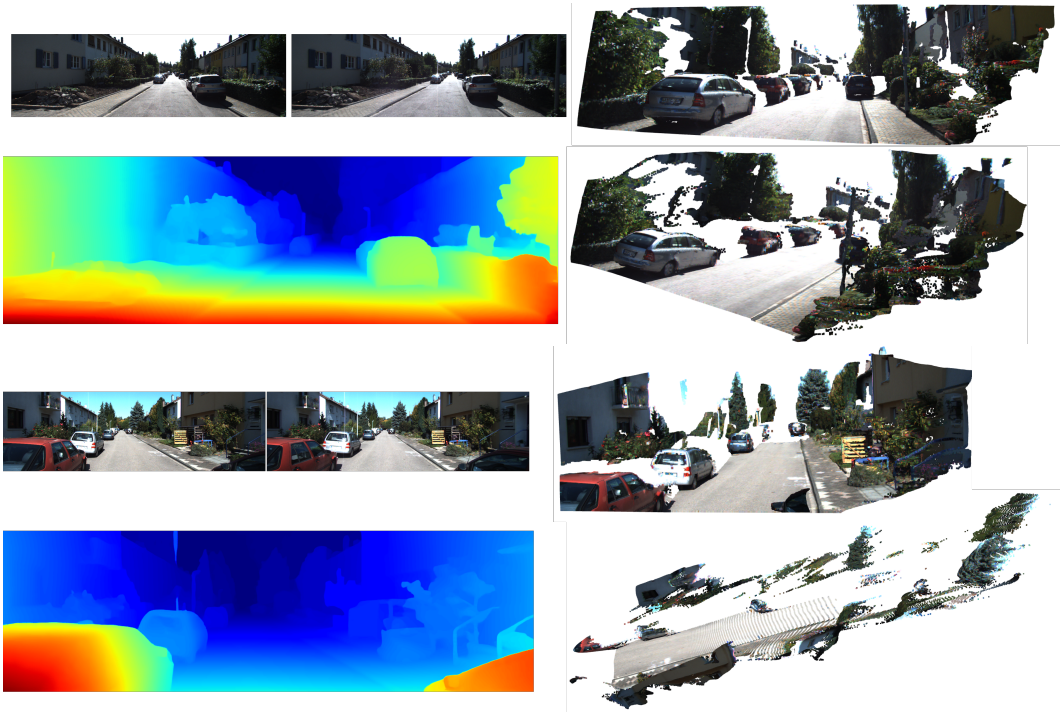


Figure 12: Depth Maps and Reprojected Point Clouds from KITTI stereo images using RAFT-Stereo

6.5 Point Cloud Registration

Once we get a pointcloud which was consistant through different frames and accurate to scale, we move to register these point clouds to get an overall 3D map of the environment seen in the video feed. For this task, we propose to use RANSAC and ICP that we had learnt in class.

We get an initial estimate of the transformation using RANSAC global registration, which is needed for the ICP algorithm. We then provide this transformation to the Colored ICP algorithm [4], to get the final estimate of the transformation. We then transform the current frame, add it to the previous frame and downsample them together. Now iterating over this, we can map the entire environment and save the pose matrices. Overall, registration of 160 frames took around 32 minutes, averaging 12 seconds per frame.

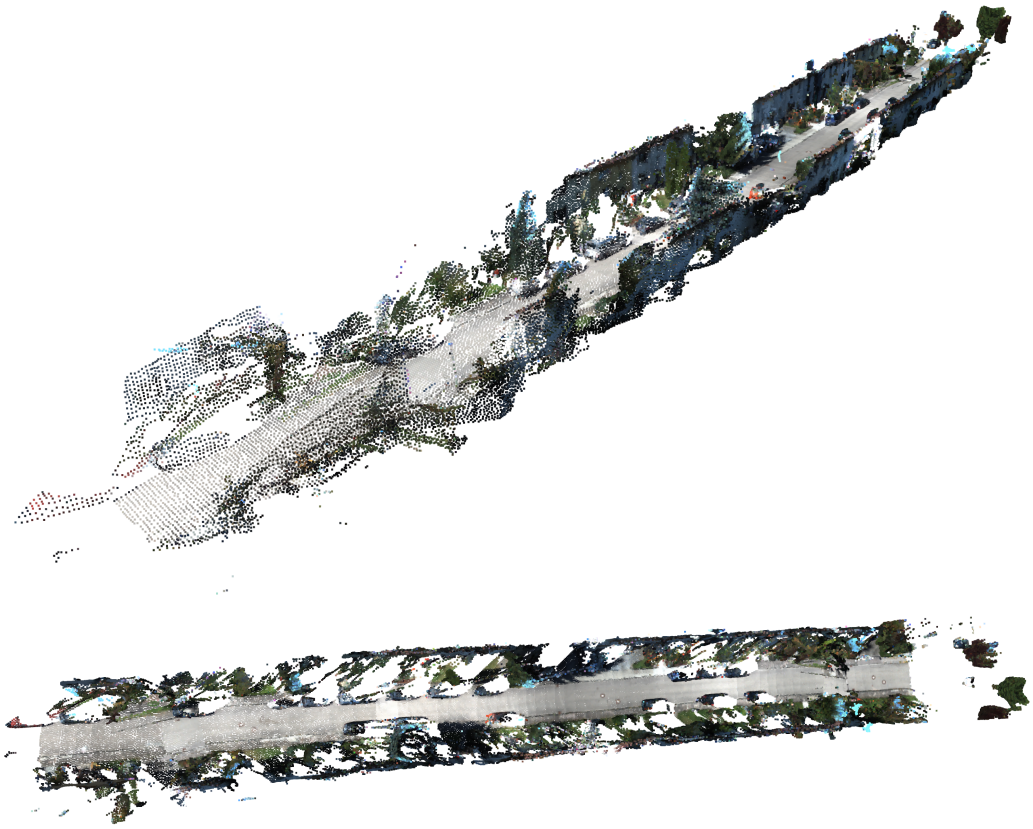


Figure 13: Point Cloud Registration results on 160 frames from the KITTI dataset. Video link.

7 Evaluation

We provide the results for different models on RMSE and a1 parameters which we have chosen for this project.

	Parameters				
Model	Mode	Image Resoultion	FPS	rmse	a1
PackNet	Monocular	640 x 192	6 - Nvidia V100	4.659	0.87
Vision Transformer	Monocular	640 x 480	30 - Nvidia 3050 Ti	2.573	0.904
RAFT Stereo	Stereo	1242 x 375	2 - Nvidia 3050 Ti	0.576	0.956

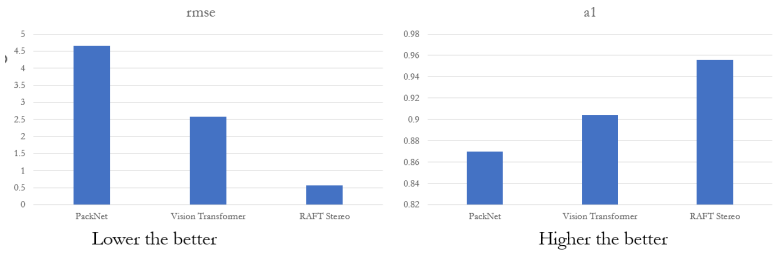


Figure 14: Evaluation Results for Depth Prediction.

8 Conclusions

From all the above experiments and evaluation, we conclude the following. For Monocular Depth Prediction using PackNet [2], which is a self supervised model did not need ground truth depth for training. PackNet could predict depth with just single camera and its calibration parameters. One of the pain points for PackNet was that the point clouds were not that accurate and the model required high compute, ran 4 frames per second during inference. MiDAS [5] on the other hand, is an unsupervised model which does not require camera calibration parameters. This helped it generalize across multiple image inputs with different contexts. These models can run in real time, but still produced point clouds which were still difficult to register. Hence we conclude that our Monocular networks were not the solution for point cloud registration.

Therefore we moved to stereo based depth prediction for which we chose RAFT-Stereo [3]. This network gave much better depth maps and hence, better point clouds. But this gain in accuracy caused an increase in compute time, which was the trade-off we chose to make. With better point clouds, we were able to map the environment by just registering the point clouds together, so this method can be applied to further problems like SLAM and 3D object detection.

References

[1] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[2] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation, 2019.

[3] Lahav Lipson, Zachary Teed, and Jia Deng. Raft-stereo: Multilevel recurrent field transforms for stereo matching, 2021.

[4] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 143–152, 2017.

[5] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction, 2021.