Methodology:

This section explains the process of generating a 3d point cloud from a single RGB image.

Depth Estimation:

We used the depth estimation module from **LeReS** to calculate the depth of the image. It is a state-of-the-art deep learning-based approach specifically designed for depth estimation. It aims to infer the depth using a single image unlike stereo or multi-view systems that require multiple images.

Architecture:

It follows a common encoder decoder architecture. The network has two output branches. The decoder outputs the depth map, while the auxiliary path outputs the inverse depth. Different losses are enforced on these two branches.

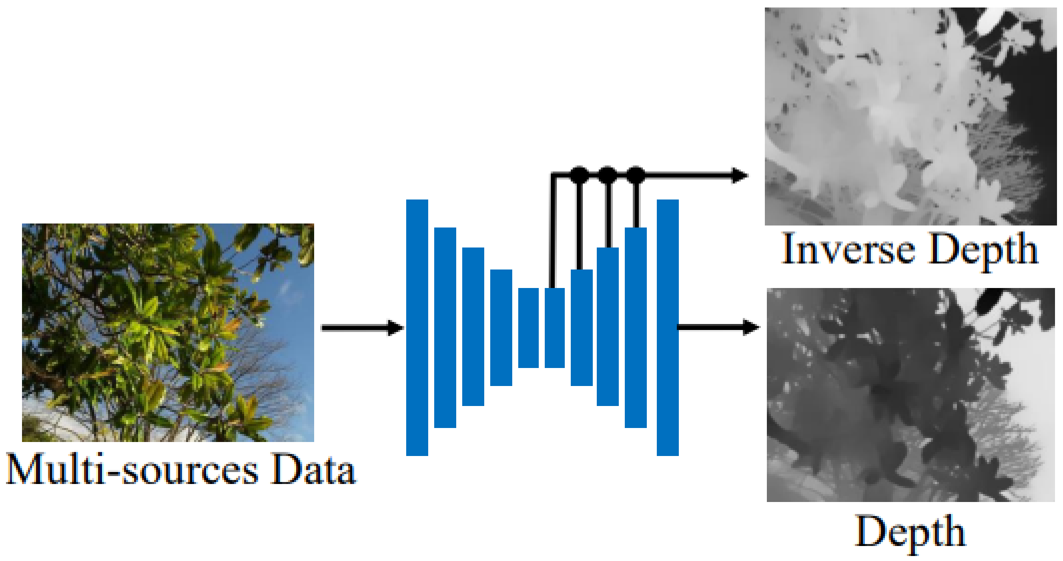


Figure 1 Network architecture for Depth Prediction model

Encoder: The encoder is based on a pretrained CNN backbone such as ResNet50 or ResNeXt101. It extracts hierarchical features from the RGB image. It reduces the dimensions of the input image while increasing the number of feature channels, capturing both low-level and high-level features.

Depth Decoder (Multi-scale Patch-Based Estimation): The decoder reconstructs the depth map from encoded features. A multi-scale depth decoding strategy refines depth prediction progressively. The network processes input patches instead of the entire image at once, which helps in reducing computational cost and improving resolution.

Attention Mechanism: A self-attention mechanism is employed to ensure global consistency. Final depth map is constructed using a fusion module that combines local and global depth information. t helps in capturing long-range dependencies between pixels, which is crucial when estimating depth in complex scenes.

Scale-Invariant Depth Prediction: LeReS inherently learns scale-consistent depth maps unlike traditional depth estimation models that require post-processing for scale alignment.

Pre-training:

We use a pretrained LeReS model with RenNeXt101 encoder. The model is trained on a variety of datasets including scenes and images from different environments. Below are the datasets used for training:

- **Taskonomy:** 114K high quality RGBD pairs from LiDAR sensor.
- **DIML:** 121K RGBD pairs of indoor scenes were used captured from calibrated stereo images.
- **3D Ken Burns:** Synthetic dataset of 51K RGBD pairs.
- **Holopix50K:** In-the -wild uncalibrated stereo images obtained from web stereo images.
- **HRWSI:** 20K RGBD images collected from diverse internet sources

Training Loss Function: LeReS uses a combination of custom loss functions to improve depth estimation accuracy across different datasets. These loss functions are designed to handle scale and shift invariance, local and global geometric consistency, and depth boundary accuracy. Following loss functions are used:

- **Image-Level Normalized Regression Loss (ILNR):** Handles dataset variations where different sources have different depth scales.
- **Pairwise Normal Loss (PWN):** Ensures local geometric consistency by enforcing normal similarity across depth edges and planar surfaces.
- **Multi-Scale Gradient Loss (MSG):** Preserves sharp depth transitions by comparing depth gradients at multiple scales.
- **Ranking Loss (For Web Stereo Data):** Used for datasets where only relative depth is available (Holopix50K, HRWSI).

The total loss function combines these components:

$$L = L_{PWN} + \lambda_a L_{ILNR} + \lambda_g L_{MSG}$$

where $\boldsymbol{\lambda_a}$ **=** 1 and $\boldsymbol{\lambda_g}$**=0.5** control the weight of each term.

Evaluation Metrics: For assessing (evaluating) the depth prediction module, LeReS uses standard depth evaluation metrics commonly used in monocular depth estimation. These metrics measure the accuracy, consistency, and quality of the predicted depth maps against the ground truth.

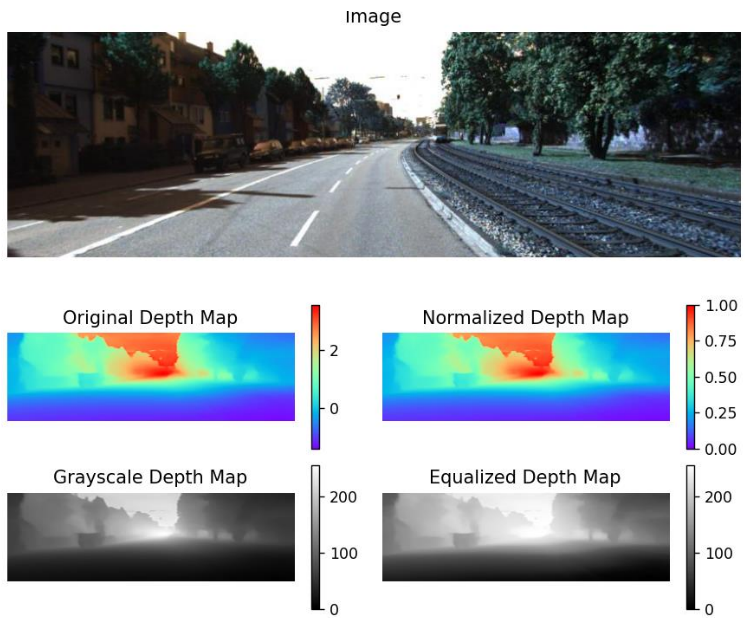
- **Absolute Relative Error (AbsRel):** Measures the absolute difference between predicted and ground truth depth, normalized by ground truth depth.
- **δn Accuracy (Threshold-Based Accuracy):** Measures the percentage of predicted depth values within a certain ratio of the ground truth.
- **Weighted Human Disagreement Rate (WHDR):** Measures how often the predicted depth ranking between two points disagrees with human annotations.
- **Locally Scale-Invariant RMSE (LSIV):** Used for evaluating 3D shape quality from depth predictions.
- **Depth Boundary Error (DBE):** Measures the accuracy of depth edges (important for object boundaries).
- **Planarity Error (PE):** Measures how well planar regions (like walls, floors) are preserved in depth maps.

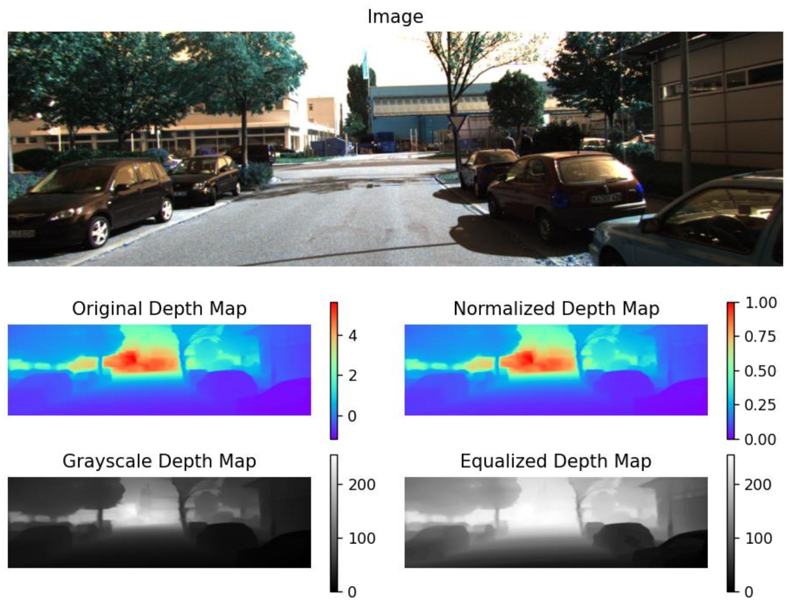
Post Processing:

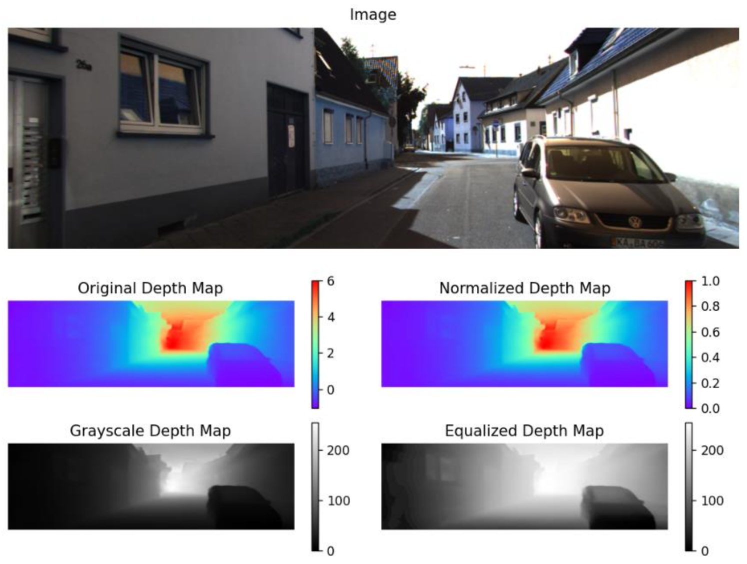
- **Normalization:** Normalization ensures all depth values fall within a consistent range (e.g., [0,1] or [0,255]).
- **Grayscale depth:** Depths are converted into grayscale images for easy visualization and processing.
- **Histogram Equalization:** Enhances contrast in the depth image by redistributing pixel intensities

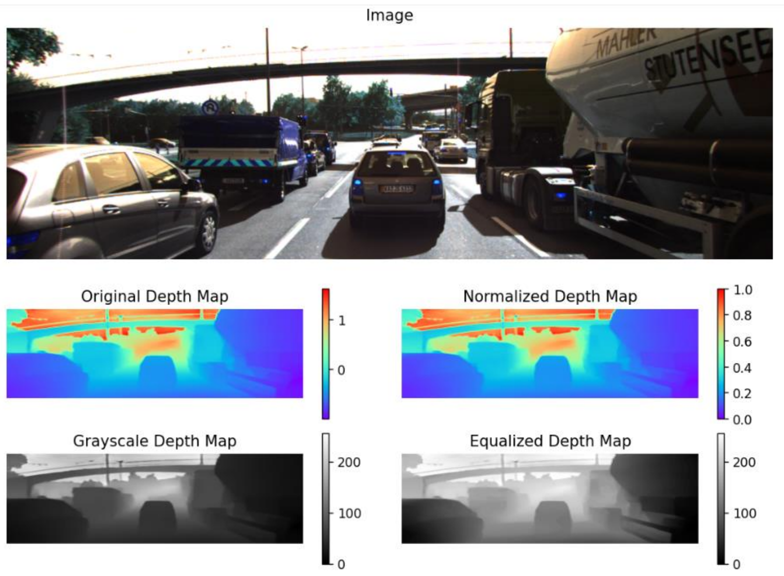
Results:

Below are some samples of the depth estimation from kitti dataset









Point Cloud Generation:

After the depth estimation, point cloud is generated using the original RGB image and the calculated depth map. Back-Projection method is used to generate 3d point cloud from a depth map and an RGB image. This method converts 2d pixels coordinates from the depth map into the 3d world coordinates using the intrinsic camera parameters.

Back-Projection: Each pixel in the depth image corresponds to a 3D point in space. The back-projection process "undoes" the camera's perspective projection by mapping the 2D pixel coordinates back into the 3D space using the known depth value.

Camera Intrinsics: Camera intrinsic parameters define the relationship between 2D image coordinates (pixels) and 3D world coordinates for a given camera. They are crucial for depth-to-3D conversion, 3D reconstruction, and computer vision tasks**.** The intrinsic matrix **K** is typically represented as:

$$K = \begin{matrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{matrix}$$

Where:

- $\boldsymbol{f_x, f_y}$ → Focal lengths in pixels (related to field of view & sensor size).
- $\boldsymbol{c_x, c_y}$ → Principal point (optical center of the image).

Since the camera intrinsics are not present, we make an estimation of the values.

Mathematical Formulation:

The back-projection formula to compute 3D coordinates **(X, Y, Z)** from depth **d** is:

$$X = \frac{(u - c_x).d}{f_x}$$

$$Y = \frac{(v - c_y).d}{f_y}$$

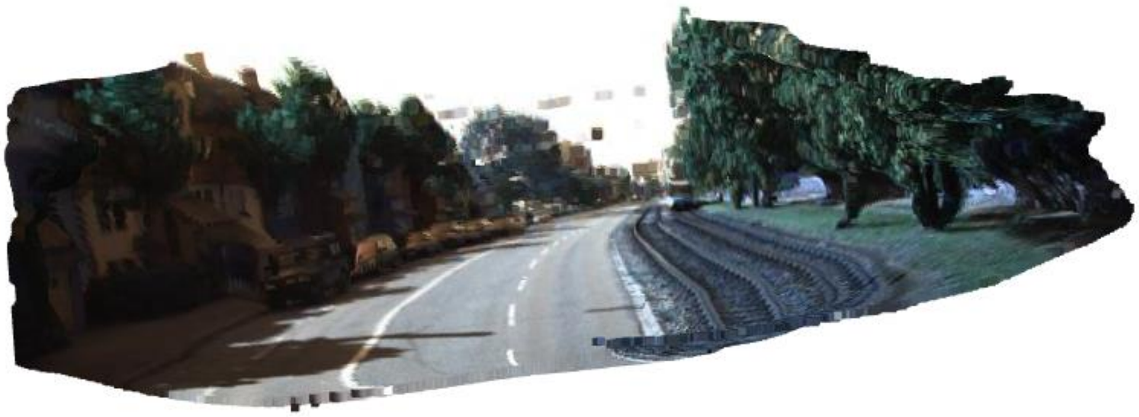
$$Z = d$$

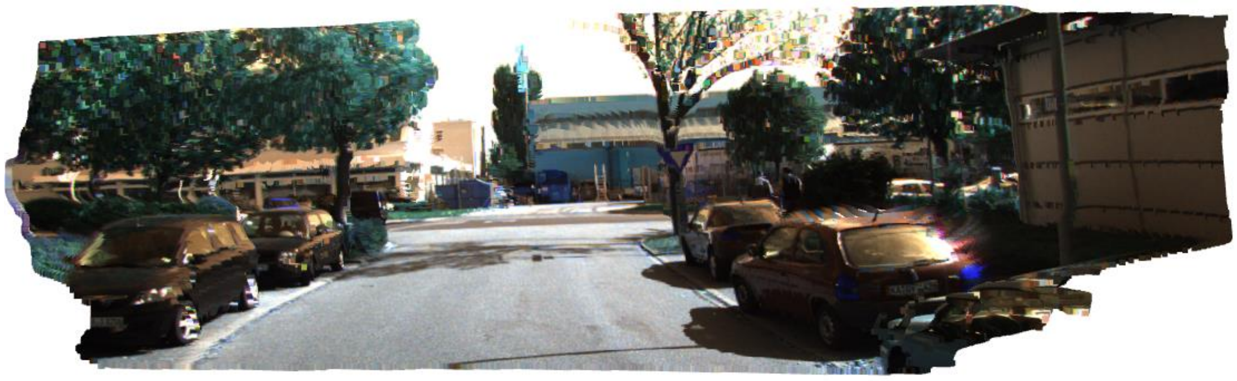
Where,

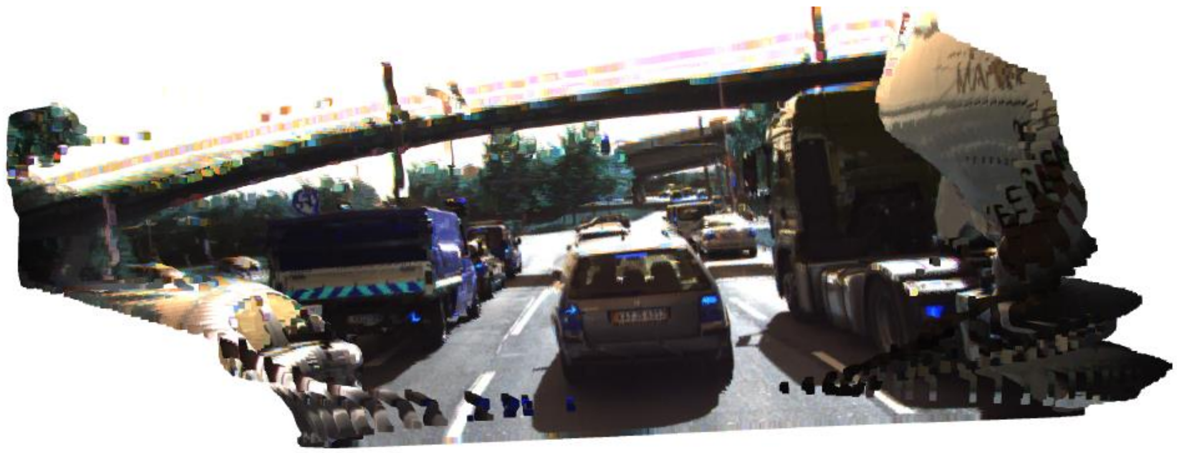
(u, v) → Pixel coordinates in the image.

d → Depth value at (u, v) (measured in meters after scaling).

Below are some results of this method:







Refinement:

Keypoint detection: Keypoint detection involves detecting distinctive feature points in an image that are invariant to scale, rotation, and illumination changes. These keypoints are used for feature matching, depth estimation, and 3D reconstruction.

SuperPoint: SuperPoint is a fully convolutional neural network that simultaneously detects keypoints (interest points) and computes their descriptors (feature vectors) in an end-to-end manner. The key innovation of SuperPoint is its ability to learn both keypoint detection and descriptor extraction in a single, unified framework, which makes it highly efficient and effective.

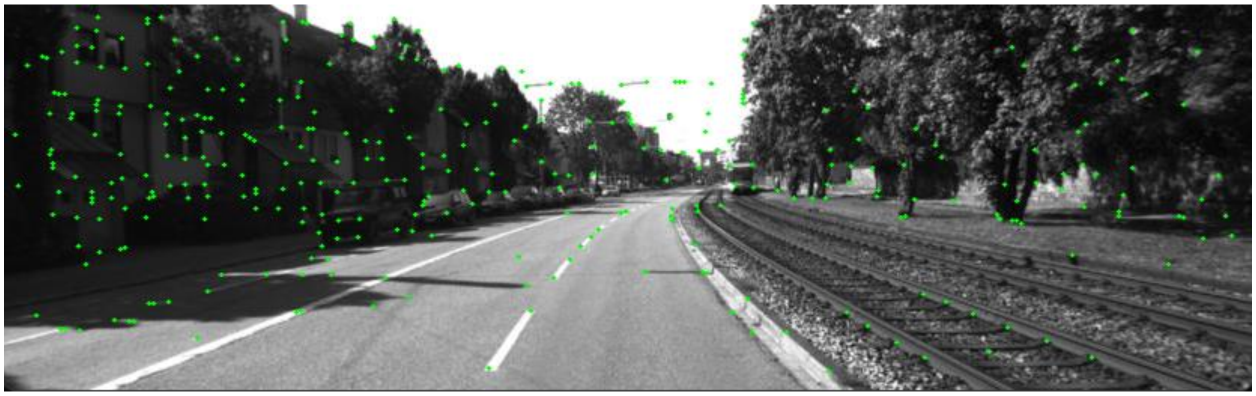
Architecture:

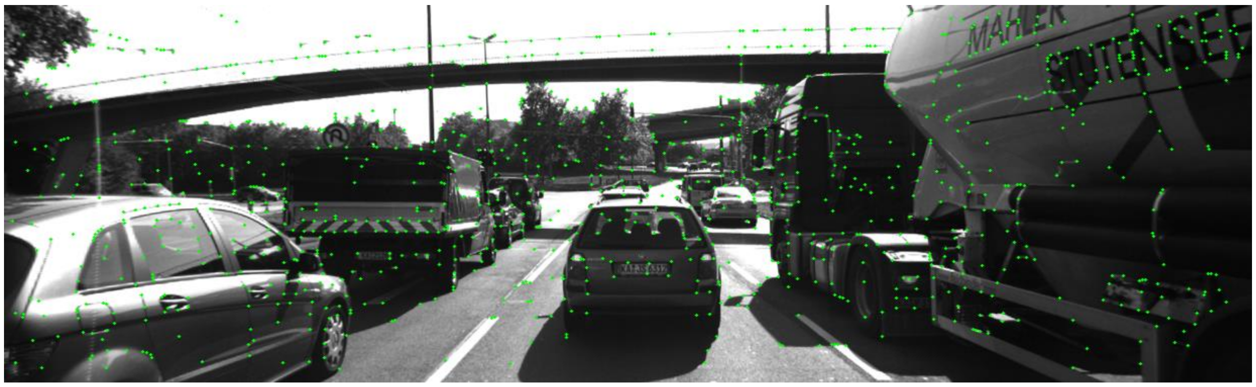
Shared Encoder: The encoder is a VGG-style convolutional neural network that processes the input image, and extracts feature maps at multiple scales. This shared encoder is used for both keypoint detection and descriptor extraction, making the model efficient.

Keypoint Decoder (Interest Point Detector): The keypoint decoder takes the feature maps from the encoder and predicts a probability map indicating the likelihood of each pixel being a keypoint. A non-maximum suppression (NMS) step is applied to select the most prominent keypoints.

Descriptor Decoder: The descriptor decoder generates a dense descriptor map for the entire image. Each pixel in the map corresponds to a feature vector (descriptor) that encodes the local appearance around that pixel. During inference, descriptors are extracted at the locations of the detected keypoints.

Results:

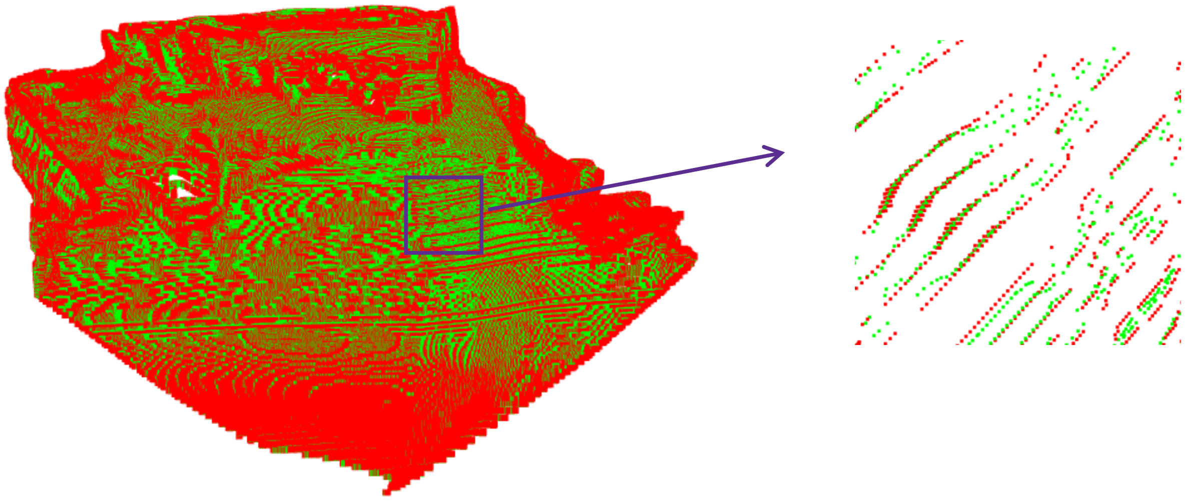




Green dots represent the detected keypoints

Denoising:

Using Keypoints: Using these keypoints and depth map, the point cloud is enhanced. More specifically, it is more smoothed. The 2d keypoints are lifted to 3d and KNN (K-Nearest Neighbour) is applied to the points. For each point, it replaces its position with the mean of



its neighbors, effectively smoothing out noise.

Figure 2 In first image, red color is the original cloud and green is the enhanced point cloud

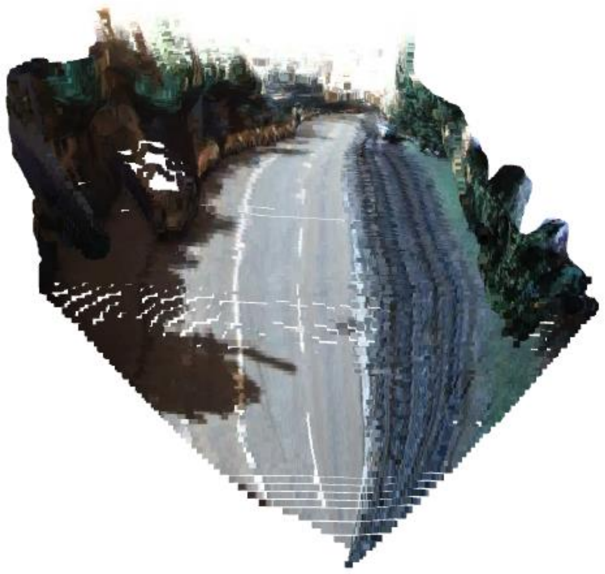


Figure 3 Enhanced Point Cloud

Statistical Method: Another method that we tried was the using the statistical outlier removal technique. The remove statistical outlier method is used to denoise a point cloud by removing points that are far from their neighbors, which are likely to be outliers.

It first computes the mean distance between the points and its neighbours which are calculated using KNN. If a point's mean distance is greater than a certain threshold (standard deviation multiplier), it is classified as an outlier and removed.

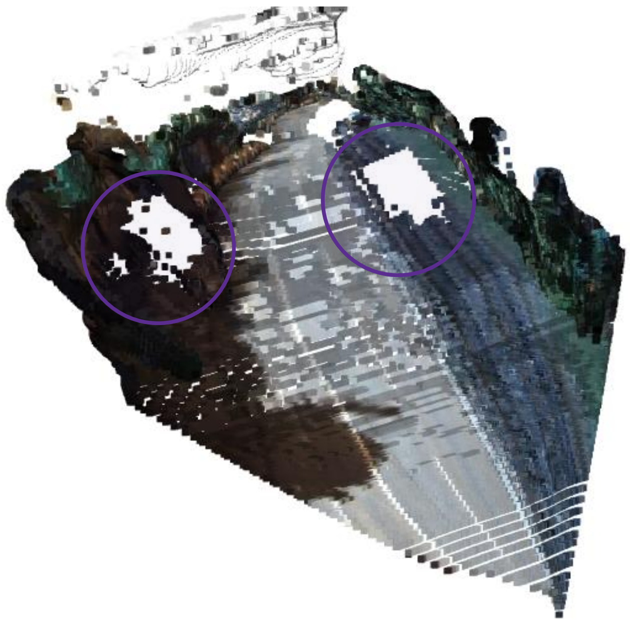


Figure 4 Outlier removal results