DYNAMIC VIEW SYNTHESIS IN UNSTRUCTURED FIELD ENVIRONMENTS

BY

YIXIAO FANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science in Electrical and Computer Engineering
in the Undergraduate College of the
University of Illinois Urbana-Champaign, Spring 2024

Urbana, Illinois

Advisor:

Professor Katherine Driggs-Campbell

ABSTRACT

The method of generating new views of a scene from novel perspectives, known as view synthesis, has seen remarkable progress thanks to advances in computer vision, machine learning, and graphics. This technology is increasingly being applied in various fields, including autonomous vehicles and robotics. For instance, robots equipped with view synthesis capabilities can be deployed in hazardous or inaccessible environments for inspection and surveillance tasks. Synthesizing views from limited input images can provide comprehensive visual information to make informed decisions without being physically present. This senior thesis delves into the development and importance of dynamic view synthesis techniques, focusing on their critical role in robotics, especially for enhancing the video frame rate during the teleoperation of agricultural robots. This application is crucial for remote operators to make timely and informed decisions. A new method is proposed that merges existing dynamic view synthesis techniques with monocular depth estimation to generate more accurate predictions under low communication bandwidth settings. Predicted images are fed to a remote operator to increase the frame rate of the video feed. Initial results on a large offline dataset collected from the field suggest the technique's viability, and additional real-world robot experiments are planned for the near future to test its real-time operational capability.

Subject Keywords: Agricultural Robots; Teleoperation System; Computer Vision; View Synthesis; Monocular Depth Estimation

To the ones who love me, for their unrequited support.

ACKNOWLEDGMENTS

I would like to extend my deepest gratitude to my advisor, Professor Katherine Driggs-Campbell, and my graduate student mentor, Neeloy Chakraborty. Their expertise, understanding, and patience, added considerably to my undergraduate experience. With their assistance, I was able to recognize my interest in robotics and artificial intelligence at an early stage and lead to this thesis to conclude my undergraduate study. Their guidance was invaluable in the planning, conduct, and analysis of my research, and the insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. Their willingness to give their time so generously has also been very much appreciated.

I am also profoundly grateful to the group of graduate students with whom I have shared the journey of my undergraduate study within the Human-Centered Autonomy Lab. Working alongside Weihang Liang, Ye-Ji Mun, Andre Schreiber, Tianchen Ji, Haonan Chen, and others has been an invaluable aspect of my experience. Their support and care have made the lab a space where I felt welcomed and valued. Looking ahead, I am eager to explore opportunities for future collaboration with them and continue contributing to the field of robotics.

Finally, I want to express my gratitude to students of the Distributed Autonomous Systems Laboratory at UIUC, especially Jose Cuaran, Arun Narenthiran Sivakumar and Mateus Valverde Gasparino for providing the dataset used for model training.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Motivation

The term teleoperation refers to the remote control of machines or systems from a distance, where the operator is not physically present at the site of the machine. This technology is typically used in environments that are hazardous, inaccessible, or too remote for direct human intervention, such as deep-sea exploration, space missions, or robotic surgery. Teleoperation systems often include cameras and sensors to provide the operator with real-time feedback, enhancing control and interaction despite the physical separation.

Teleoperation systems can also be deployed in agricultural field environments to control agricultural robots. However, similar to other applications, teleoperation systems in fields suffer from significant latency. Latency typically occurs for the data transmission between robots and the base station. For instance, images transmitted back from the robot can be delayed, and the operator's command cannot be processed by the robot timely. The reduction of latency can lead to improvements in both user experience and safety considerations. Especially when agricultural robots are deployed in dynamic and complex field environments, latency can introduce significant risks, undermining the operator's ability to make timely decisions and react to unforeseen events. Where the robots travel through rows of stalks, large-scale leaves and branches can partially block the camera. With significant latency, the operator will fail to prevent accidents or damage, posing a direct threat to the safety of both the operational environment and the robot itself. Therefore, addressing latency is essential not only for enhancing the efficiency and reliability of teleoperated systems but also for ensuring the safety and success of missions in complex, unstructured field environments.

As such, one direction of works rely on view synthesis to address this issue. View synthesis is a technique in computer graphics and vision that involves generating new images of a scene from viewpoints that were not originally captured by cameras. This process uses existing images or video frames along with depth information or geometric data about the scene to interpolate or extrapolate new views. View synthesis techniques can be used to increase the frame rate of video sequences at the base station. After the updated robot pose is transmitted back to the base station, we can immediately predict the updated frame without waiting to receive it from the robot, thereby improving the effectiveness of monitoring and manipulating field robots.

Predicting future frames in a video sequence conditioned on past images is challenging in unstructured field environments. Recent work on video prediction achieves excellent performance by leveraging the use of geometrical information of the environment. SynSin [1], a view synthesis model, is commonly used to extract the 3D geometry of the world from a single image. However, verified thorough experiments, training the SynSin model on images collected in fields does not provide promising results as expected, partially due to complex and unstructured features from field environments. Therefore, we searched for other geometry-based methods and landed on the work by Nagabhushan et al.[2], where the authors apply optical flow and multi-plane images to accurately predict object motion, eventually assisting in forming future frames. Nevertheless, their work is only evaluated on data collected from simulators, which implies the availability of accurate depth maps that are not presented in field environments. Thus, additional work can be done to produce accurate depth map by using techniques such as monocular depth estimation.

1.2 Contributions

We present the following contributions:

- We refine an existing framework of dynamic view synthesis to be applied in unstructured field environments.
- We build upon the state-of-art monocular depth estimation model, developing a more efficient framework to estimate metric depth in unstructured field environments, and incorporate it within the view synthesis model.

2 RELATED WORKS

2.1 Video Prediction

In recent years, learning-based strategies for video prediction have demonstrated significant promise, as evidenced by a series of studies [3, 4, 5]. Within this domain, a considerable portion of research has achieved success in the specific area of temporal video prediction. This concept revolves around the prediction of future video frames over a brief timeframe. For instance, research indicated by Villegase et al.[5] introduces methodologies that prioritize predicting the overarching structure of future frames rather than their individual pixels, yielding encouraging outcomes.

However, a common limitation among these studies is the presumption of a static background, implying that objects within the scene remain motionless over short intervals. To address the challenges posed by dynamic backgrounds, certain approaches have adopted optical flow techniques [6]. This method allows for the separate estimation of pixel velocities for both dynamic and static objects, facilitating the construction of predicted frames. Yet, this strategy is limited to considering the movement of pixels across the 2D plane of the image, neglecting the 3D geometry of the scene.

In addition to methods based on optical flow, some researchers have explored geometry-based frameworks for temporal video prediction [7]. These methodologies typically leverage depth data, RGB frames, and accurate predictions of camera movements in the future. Such approaches share similarities with the objectives of view synthesis. However, in contrast to temporal view synthesis, which employs both camera motion and depth information, temporal video prediction models often exclude these critical elements. Instead, the focus of temporal view synthesis is on utilizing camera movements to specifically predict the local motion of objects, distinguishing it from the broader aims of temporal video prediction.

2.2 View Synthesis

View Synthesis is the process of creating an image from a new viewpoint using one or more images from different viewpoints. These models usually require the camera's position to be known, while the depth remains to be determined. Depth can be learned either by directly estimating it or through indirect methods like Multiplane Images (MPI) [8] or the more recent Neural Radiance Fields (NeRF) [9] techniques, which have shown promising results in view synthesis. Depth Image Based Rendering (DIBR) [10] models, on the other hand, operate under the assumption that depth information is available, using a warp-and-infill strategy to address disocclusions, with some methods focusing on removing foreground objects to reconstruct and infill the background before applying motion compensation.

Various strategies have been explored for temporal view synthesis, which deals with static scenes, and dynamic view synthesis, which focuses on creating video frames of a dynamic scene from a new viewpoint. These methods include leveraging both single-view and multi-view depth information [11], employing static and dynamic versions of NeRF [12], and utilizing MPI representations. However, these methods generally assume the scene does not change between views and do not account for object motion.

Somraj et al. combine the idea of optical-flow based methods of video prediction and recent work of view synthesis to construct a novel dynamic view synthesis model by decoupling camera and object motion [2]. However, their work assumes that ground-truth depth information is available at every time step, which is not a valid assumption for most robotics applications. Especially in extremely unstructured environments, depth information captured by cameras is usually corrupted.

2.3 Image and Video Inpainting

View synthesis methods usually face the challenge of generating unseen parts in the 3D world when the camera moves to new poses. In the setting of view synthesis, the step of warping a current frame image to a new image based on future camera pose and depth information is crucial. However, the image after warping may contain holes because some regions become unobstructed in the new frame. Kanchana et al. show that the disoccluded pixels can be infilled by copying intensities from around pixels given that temporal view synthesis considers closely adjacent frames [13].

2.4 Monocular Depth Estimation

2.4.1 Single-Image Depth Estimation (SIDE)

Supervised techniques for inferring depth from a single image generally fall into two categories: those that estimate metric depth [14, 15] and those focused on relative depth [16]. Metric depth models, often trained on specific datasets, tend to be more prone to overfitting and usually struggle to adapt to new settings or varying depth scales. Conversely, relative depth models, trained across a broader range of datasets with annotations of relative depth and employing scale-invariant losses, boast greater adaptability. However, their utility is limited in scenarios requiring precise metric depth information, as they yield depth estimates without a specific scale or shift.

In a novel approach, Bhat and colleagues have merged the methodologies of these two approaches to create a depth estimation model named ZoeDepth [17]. This model stands out for its superior ability to generalize across different environments while preserving metric scale accuracy. However, the model's complexity leads to prolonged inference times, rendering it impractical for applications needing to enhance frame rates in real-time.

2.4.2 DINOv2

DINOv2 [18], building on the self-supervised learning framework of its predecessor, DINO [19], represents the state-of-the-art for monocular relative depth estimation. While not specifically designed for depth estimation, DINOv2's sophisticated learning framework offers valuable tools for improving accuracy and generalization in monocular depth estimation models. By leveraging its advanced capabilities in learning rich visual representations without labeled data, DINOv2 could significantly improve feature extraction, making it adept at identifying depth-related cues from single images. Its potential for self-supervised learning addresses the challenge of scarce labeled depth data, allowing for effective scaling of training processes. Furthermore, the transferability of the learned features to depth estimation tasks, possibly augmented by attention mechanisms, could enhance model performance across varied environments and conditions. However, with its limitation of relative depth estimation, the model along can hardly be applied to our application. Similar to ZoeDepth, it suffers from the problem of long inference time to be run in real-time.

3 BACKGROUND

3.1 Problem Formulation

The task of applying dynamic view synthesis for frame rate upsampling by k times can be formulated as below:

- Given previous frames $\{f_{n-ak}, ..., f_{n-k}, f_n\}$, corresponding depth maps $\{d_{n-ak}, ..., d_{n-k}, d_n\}$, extrinsic camera poses $\{\mathbf{T}_{n-ak}, ..., \mathbf{T}_{n-k}, \mathbf{T}_n\}$, camera intrinsic matrix $\mathbf{K}$, and future extrinsic camera poses $\{\mathbf{T}_{n+1}, \mathbf{T}_{n+2}, ..., \mathbf{T}_{n+k-1}\}$, predict $\{f_{n+1}, f_{n+2}, ..., f_{n+k-1}\}$. a represents the number of past frames we consider.

- In this thesis, by considering the computational constraint in real-time, we simplify this problem formulation by setting $a = 1$ and $k = 2$. Thus, the task becomes: Given previous frames $\{f_{n-2}, f_n\}$, corresponding depth maps $\{d_{n-2}, d_n\}$, extrinsic camera poses $\{\mathbf{T}_{n-2}, \mathbf{T}_n\}$, camera intrinsic matrix $\mathbf{K}$, and future extrinsic camera poses $\{\mathbf{T}_{n+1}\}$, predict $\{f_{n+1}\}$.
  - For the problem formulation we define, the previous frame f_{n-1} is not given as it is a frame that should be predicted by the model at the previous time step.

3.2 Temporal View Synthesis

We now describe the problem formulation of temporal view synthesis. For the basic task of temporal view synthesis, we assume the scene is static and objects are stationary, which means motion in the video only comes from egomotion (camera motion).

- Given the current frame f_n, the corresponding depth map d_n, and transformation matrix of the camera $\mathbf{T}_n$ from f_n to f_{n+1}, the goal is to predict the frame f_{n+1}.

- To predict f_{n+1}, we can warp homogeneous pixel coordinates in f_n to f_{n+1} by using the transformation matrix and depth map:

$$\mathbf{p}^w_{n+1} = \mathbf{K}G\{\mathbf{T}_n F\{d_n(\mathbf{p}_n)\mathbf{K}^{-1}\mathbf{p}_n\}\} \tag{3.1}$$

  $\mathbf{p}_n$ is a 2D pixel homogeneous location of the frame f_n. $\mathbf{K}$ is the camera intrinsic matrix. Functions F and G represent the transformation from 3×1 to 4×1 vectors and vice versa. By performing this operation, pixel intensities in f_{n+1} can be estimated for regions that can be mapped to f_n. Besides these known regions, there are other pixels which cannot be interpolated from f_n, due to the fact that those regions in the 3D world are not in the view of the camera in f_n.

For our problem statement we cannot directly apply this solution of view synthesis since it is invalid to assume objects in agricultural fields are static.

3.3 Multi-Plane Images (MPI)

Our work is based on the dynamic view synthesis model proposed by Somraj et al. [2], in which the authors apply the idea of MPI representation, originally introduced by Zhou et al. [20], when estimating optical flow.

Zhou et al. introduced the concept of expanding a 2D RGB frame into a multi-plane image consisting of several RGBA planes positioned at varying depths. In each plane, the alpha channel (α, ranging from 0 to 1) indicates the presence of scene elements at that specific depth. Somraj etc. generate the MPI directly using an RGB-D image, leveraging the depth information it provides. To do this, they uniformly select Z planes across the inverse depth range from the scene's minimum to maximum depths. At every pixel location $\mathbf{x}$, they assign $\alpha = 1$ to the plane that is closest to the actual depth of $\mathbf{x}$, while setting $\alpha = 0$ in all other planes, effectively creating a one-hot vector of α values at each location $\mathbf{x}$. We enhance the MPI format to include actual depth values in an extra channel, alongside the existing

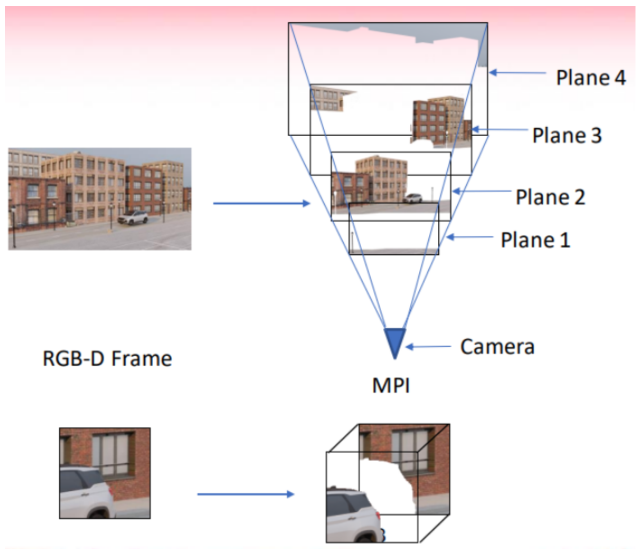

Figure 3.1: Example of MPI representation [2]

RGBA channels. Thus, the MPI format is represented as $m_n = \{c_n, d_n, \alpha_n\}$, where c_n, d_n, and α_n correspond to the RGB, depth, and alpha channels, respectively. For warping an MPI to a different camera perspective, Somraj et al. utilize reprojection and bilinear splatting techniques [13] similar to the one used by temporal view synthesis discussed in 3.2. Lastly, to compose a 2D frame from an MPI, they apply alpha compositing in a back-to-front sequence using the established over operation [20].

4 METHODOLOGY

4.1 Dynamic View Synthesis Model

We adapt the work of Somraj et al. [2] to implement dynamic view synthesis by generally following the approach described below.

As mentioned in Section 3.1, initially we are given the RGB images $\{f_{n-2}, f_n\}$, camera intrinsic matrix $\mathbf{K}$, camera poses $\{T_{n-2}, T_n, T_{n+1}\}$, and depth maps $\{d_{n-2}, d_n\}$. The goal is to predict f_{n+1}.

Algorithm 1 Dynamic View Synthesis Algorithm

```
$m_n \leftarrow MPI(f_n, d_n)$
$m_{n-2} \leftarrow MPI(f_{n-2}, d_{n-2})$    ▷ Create MPI representations for $f_n$ and $f_{n-2}$
$m^w_{n-2} \leftarrow Warp(T_{n-2}, T_n, m_{n-2})$    ▷ Warp $m_{n-2}$ in the view of $T_n$
$\mathbf{u}_{n\to n-2} \leftarrow$ 3D optical flow between $m_n$ and $m^w_{n-2}$
$\hat{\mathbf{u}}_{n\to n+1} \leftarrow interpolate(\mathbf{u}_{n\to n-2})$    ▷ Interpolate future object motion
$\hat{m}^w_n \leftarrow Warp(T_{n+1}, T_n, m_n)$    ▷ Warp $m_n$ in the view of $T_{n+1}$
$\hat{m}_{n+1} \leftarrow \hat{\mathbf{u}}_{n\to n+1} + \hat{m}^w_n$    ▷ Combine object motion with $\hat{m}^w_n$
$f_{n+1} \leftarrow$ infilling and alpha composition from $\hat{m}_{n+1}$
```

Figure 4.1 provides a details of the model structure.

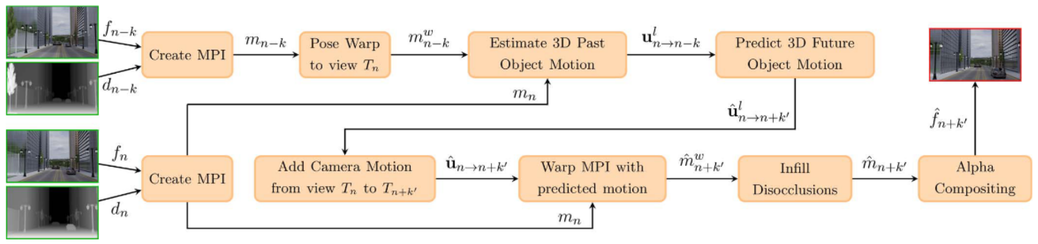
f_{n-k}
d_{n-k}
Create MPI
m_{n-k}
Pose Warp to view T_n
m^w_{n-k}
Estimate 3D Past Object Motion
$\mathbf{u}^l_{n\to n-k}$
Predict 3D Future Object Motion
m_n
$\hat{\mathbf{u}}^l_{n\to n+k'}$
$\hat{f}_{n+k'}$
f_n
d_n
Create MPI
Add Camera Motion from view T_n to $T_{n+k'}$
$\hat{\mathbf{u}}_{n\to n+k'}$
Warp MPI with predicted motion
$\hat{m}^w_{n+k'}$
Infill Disocclusions
$\hat{m}_{n+k'}$
Alpha Compositing
m_n

Figure 4.1: Flow diagram of the dynamic view synthesis model [2]

4.2 Monocular Depth Estimation Model

The dynamic view synthesis model requires accurate depth maps, which are difficult to acquire in agricultural fields, as the unstructured environment causes the cameras to record noisy, oftentimes sparse, depth data, as shown in Figure 4.3. Thus, we decide to apply monocular depth estimation models to obtain dense depth.

The aforementioned two monocular depth estimation models ZoeDepth [17] and DINOv2 [18] both have their advantages and disadvantages. The most significant issue is the inference time. For both models, the inference time on a single RGB image with size 1280×720 is around 2 seconds, using a RTX 2080 GPU, which is not feasible to run in real-time. In terms of quality of the results, ZoeDepth and DINOv2 slightly differ. DINOv2 outperforms ZoeDepth in terms of estimating depth for tiny features, given its robuts feature extraction model. However, one obvious disadvantage of DINOv2 is that it can only estimate relative depth. Comparison of their performance can be drawn visually from Figure 4.4 and 4.5.

Therefore, we decide to adapt the backbone of DINOv2 and apply some modifications to decrease the inference time, which will be further discussed in section 4.2.2. Before we can train the model, we need to have access to ground-truth depth maps, but it is impossible to obtain dense depth maps for unstructured field environments, as mentioned before. As a result, we leverage ZoeDepth and DINOv2 to build pseudo ground-truth depth maps in meters.

4.2.1 Pseudo Ground-truth Depth Map

For a single RGB image, given DINOv2, we can obtain an accurate relative depth map d_r. Relative depth map is constructed with depth values between 0 and 1 $\{0 \leq d_r(x, y) \leq 1\}$. Given ZoeDepth, we can obtain a less accurate metric depth d_m. Given the internal attributes of relative and metric depth, all the depth values in d_r should form a linear relationship with corresponding depth values in d_m if both depth maps are accurate. Thus, using a linear regression model, we can fit a linear function that relates d_r and d_m for every

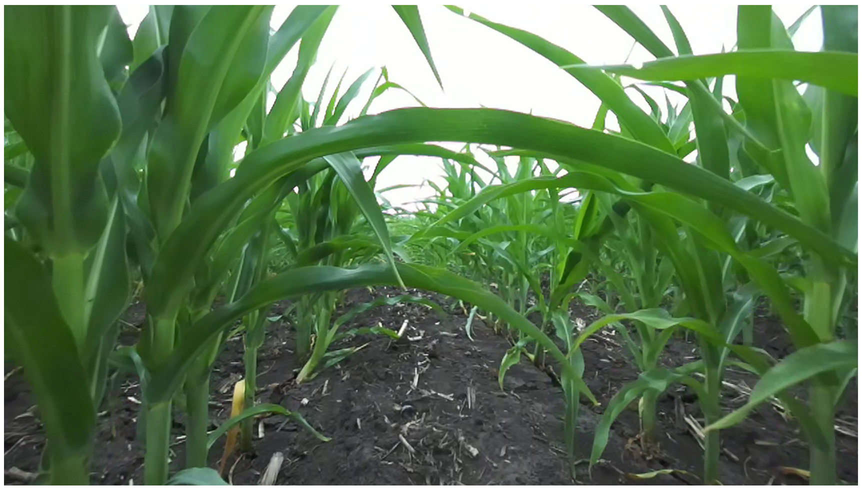

Figure 4.2: RGB image

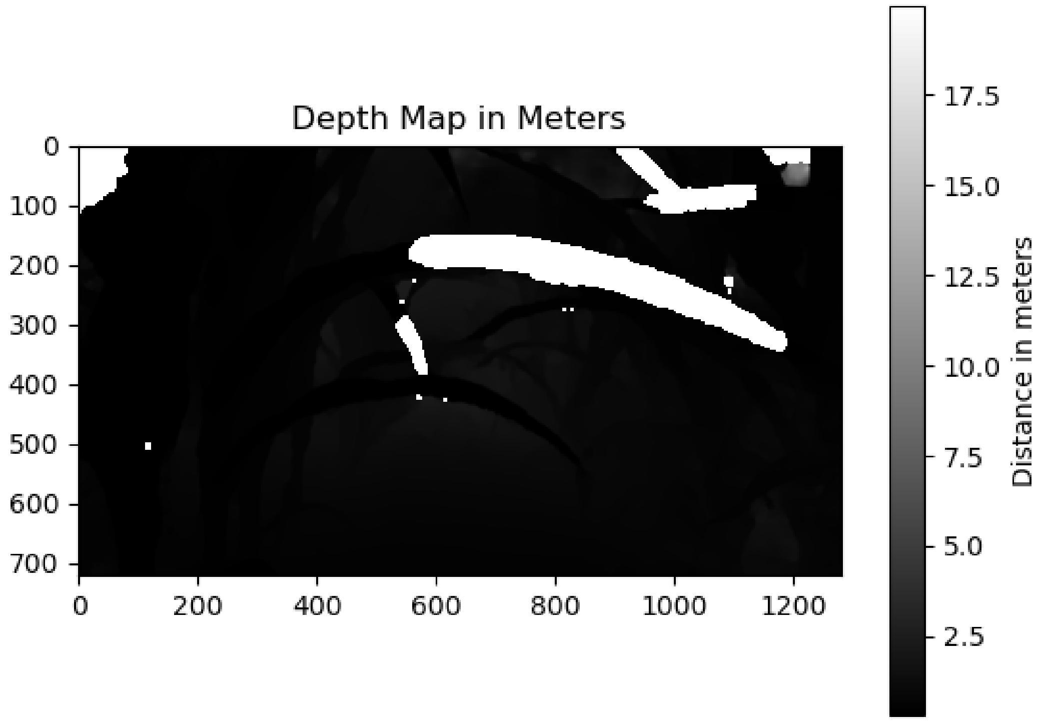

Figure 4.3: Sparse depth map from the camera; White spots are holes that cannot be predicted by the camera

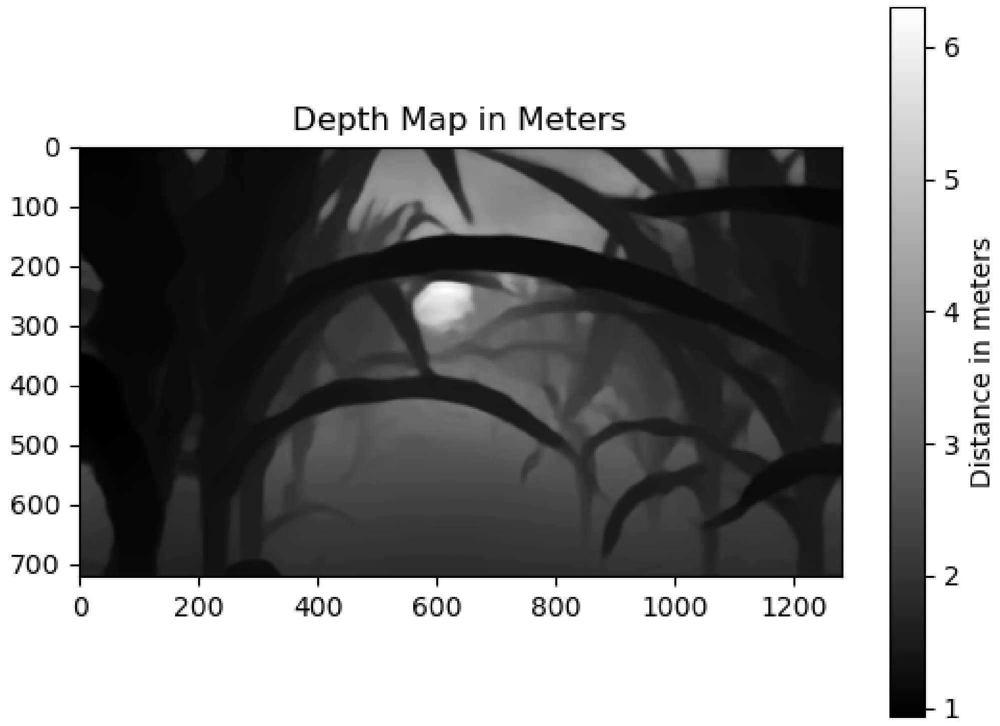

Figure 4.4: Metric depth estimation from ZoeDepth

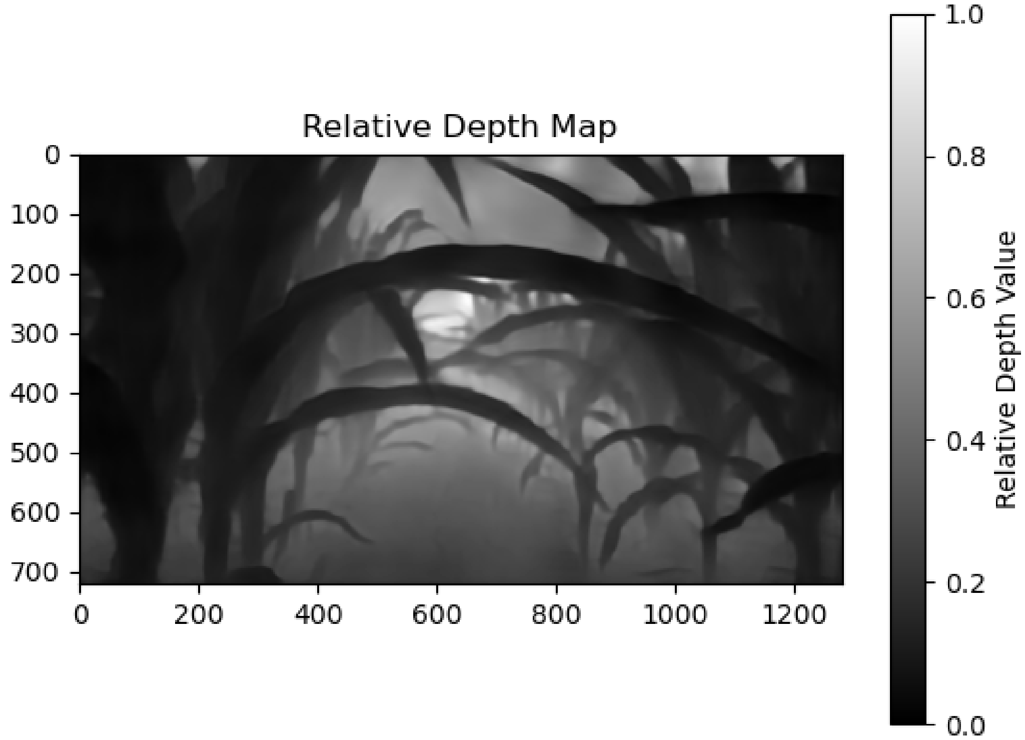

Figure 4.5: Relative depth estimation from DINOv2

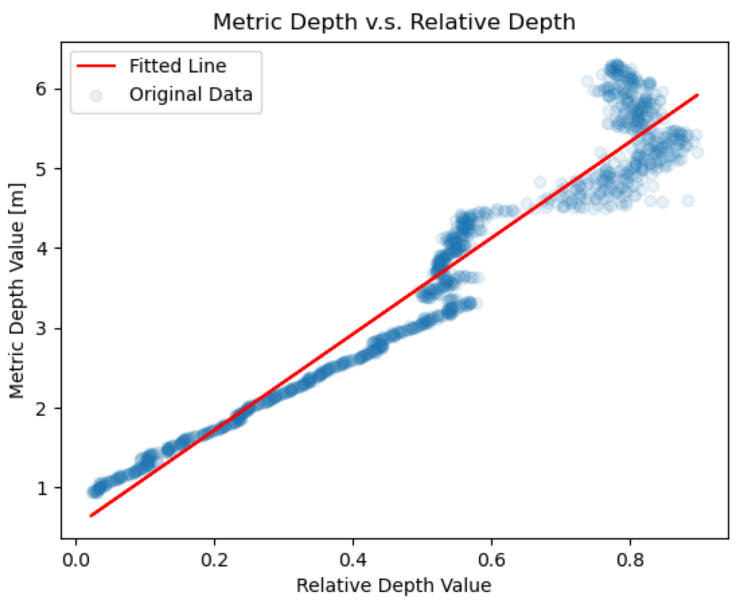

Figure 4.6: Metric depth and relative depth (DINOv2 and ZoeDepth, Avg)

RGB image in the dataset respectively:

$$d_m = f(d_r)$$

To form the final pseudo ground-truth depth map, we apply the function $f(\cdot)$ to d_r and obtain d_{pseudo}. Figure 4.8 shows an example of the result.

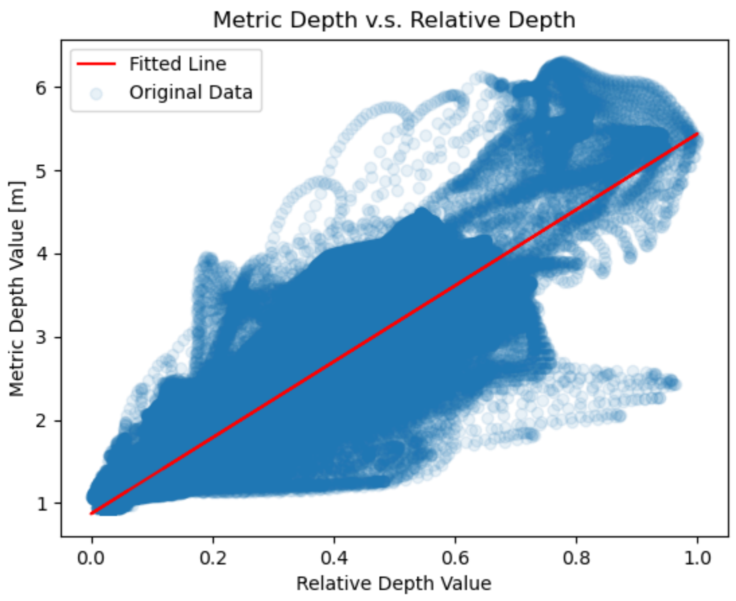

Figure 4.7: Metric depth and relative depth (DINOv2 and ZoeDepth, Raw)

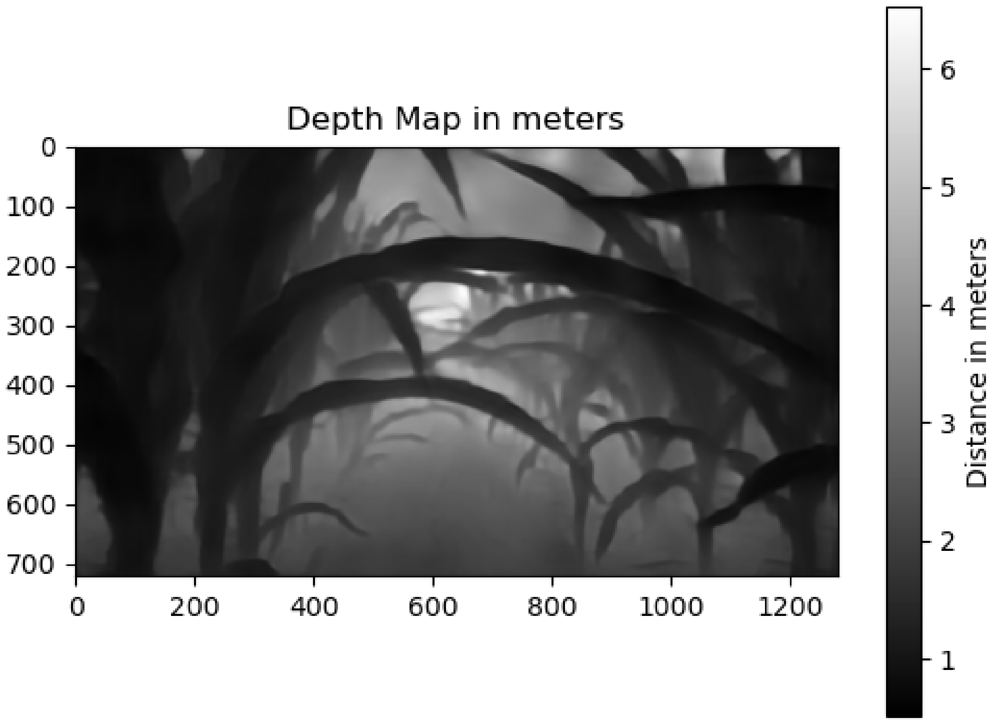

Figure 4.8: Example of the pseudo ground-truth depth map

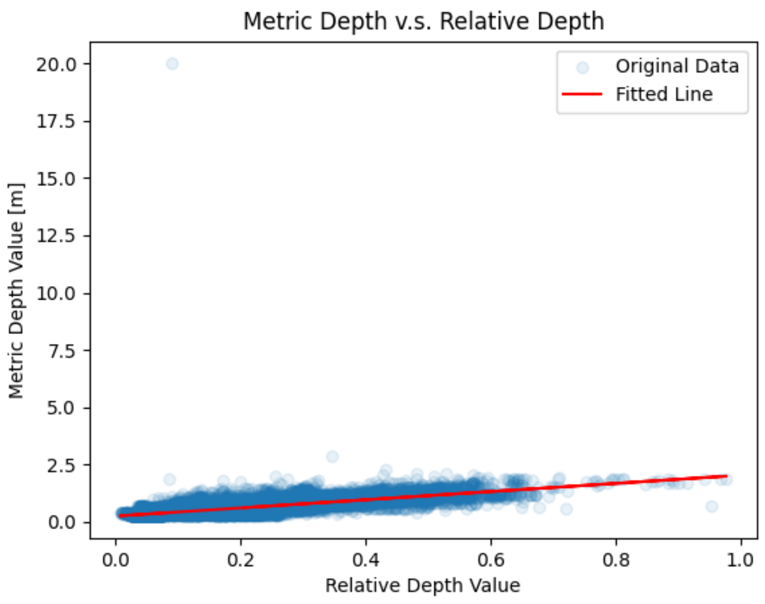

Figure 4.9: Metric depth and relative depth (DINOv2 and sparse depth, Avg)

Originally, one relative depth value can correspond to multiple metric depth values, as shown in Figure 4.7, which is foreseeable since the two depth maps are generated by two different models. We cannot guarantee there is a one-to-one mapping between relative and metric depth. As a result, the fitted line will be affected by some noisy data points. Therefore, for every relative depth, we average its corresponding metric depth values and fit a linear relationship based on that as shown in Figure 4.6.

Either using averaged or raw data, we we are unable to replace d_m, the metric depth from ZoeDepth, with d_{raw}, which is the original metric depth map collected by the depth camera. This conclusion is based on experiments that intend to fit a linear relationship between d_r and d_{raw}. Due to the sparsity and noisiness of d_{raw}, a linear function can hardly be solved. Results are shown in Figure 4.9 and Figure 4.10.

4.2.2 Modifications based on DINOv2

The general structure of DINOv2 model can be summarized as follows. The model follows the popular encoder-decoder architecture. The encoder, also known as the DINO backbone, is the key part that is used to learn numerous

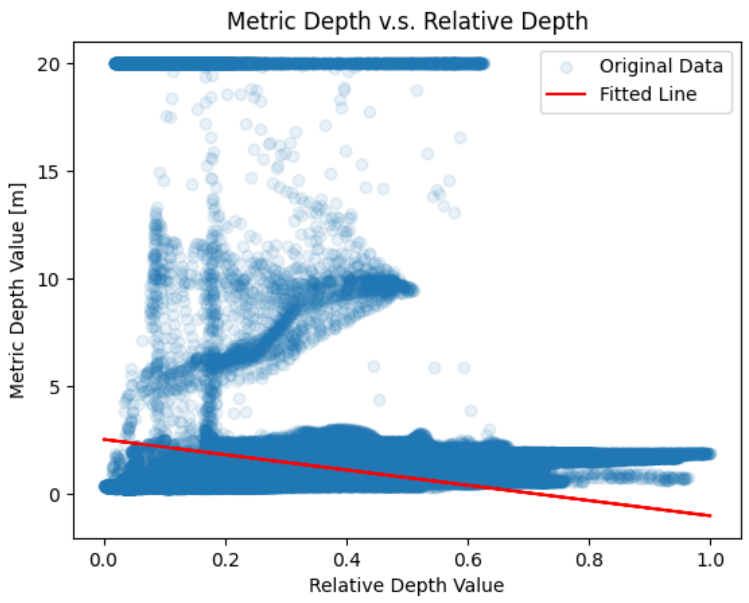

Figure 4.10: Metric depth and relative depth (DINOv2 and sparse depth, Raw)

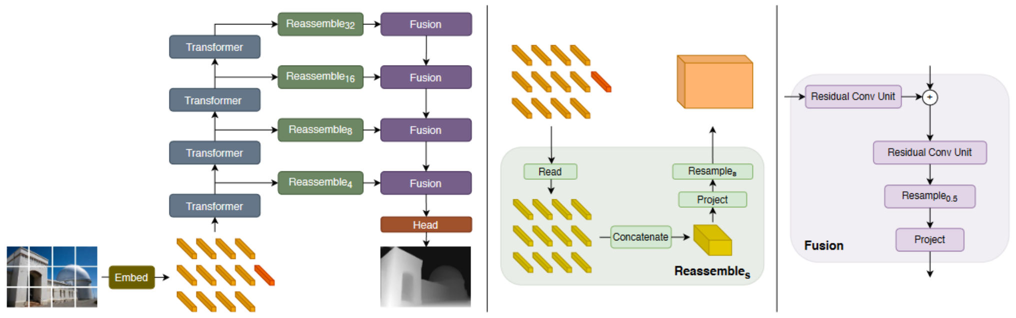

Figure 4.11: DPT model structure [21]

features from a single RGB images. The decoder, also known as the DPT head, is originally adapted from [21]. The major role of the decoder is to decode encoded features to dense image output by using a convolutional neural network.

As the DINO backbone proves to be powerful for learning features from images, we choose to keep it unchanged in our model. In addition, since one of our goals is to minimize the model inference time, we adapt the smallest pretrained backbone, *vits14*. For the decoder part, we reduce the number of convolutional layers for the *Fusion* block and modify the structure of final depth head, denoted as *Head*.

We design two distinct architectures for the depth head for two different purposes. The first task is metric depth estimation. Additionally, due to limitations of DINOv2 and ZoeDepth, both models cannot produce satisfying results when estimating depth of objects far from the camera, particularly, depth for the sky. Figure 4.12 shows the limitation to when estimating the depth for sky. Thus, the second task is to classify regions of the sky from the RGB image and assign infinity depth value afterwards.

The method to build the complete depth estimation model can be summarized as follow:

- Given the approach describe in 4.2.1, a training dataset D_1 can be constructed:

  $$D_1 = \{\mathbf{I}_{raw}, \mathbf{D}_{pseudo}, \mathbf{D}_{raw}, \mathbf{M}_{sky}\}$$

  where $\mathbf{I}_{raw}$, $\mathbf{D}_{pseudo}$, and $\mathbf{D}_{raw}$ represent sets of RGB images, pseudo ground-truth dense depth maps, and raw sparse depth maps respectively. $\mathbf{M}_{sky}$ is a binary mask indicating each pixel location can be classified as sky or not in $\mathbf{I}_{raw}$. If a pixel location (x, y) is considered as a part of the sky, $\mathbf{M}_{sky}(x, y) = 0$. Otherwise, $\mathbf{M}_{sky}(x, y) = 1$. This mask is obtained from $\mathbf{D}_{raw}$, where infinity depth values are regarded as depth values for the sky.

- Similarly, the second training dataset D_2 can be constructed:

  $$D_2 = \{\mathbf{I}_{raw}, \mathbf{M}_{sky}\}$$

- Denote the DINO backbone model as f_{dino}, the decoder backbone as $f_{decoder}$, the first depth head as f_{head1}, and the second depth head as f_{depth2}. Thus, we can represent the first model for training:

  $$f_1 = \{\bar{f}_{dino}, f_{decoder}, f_{head1}\}$$

  A bar is associated with the DINO model notation to represent the weights are fixed. The same notation will be used onwards. The model

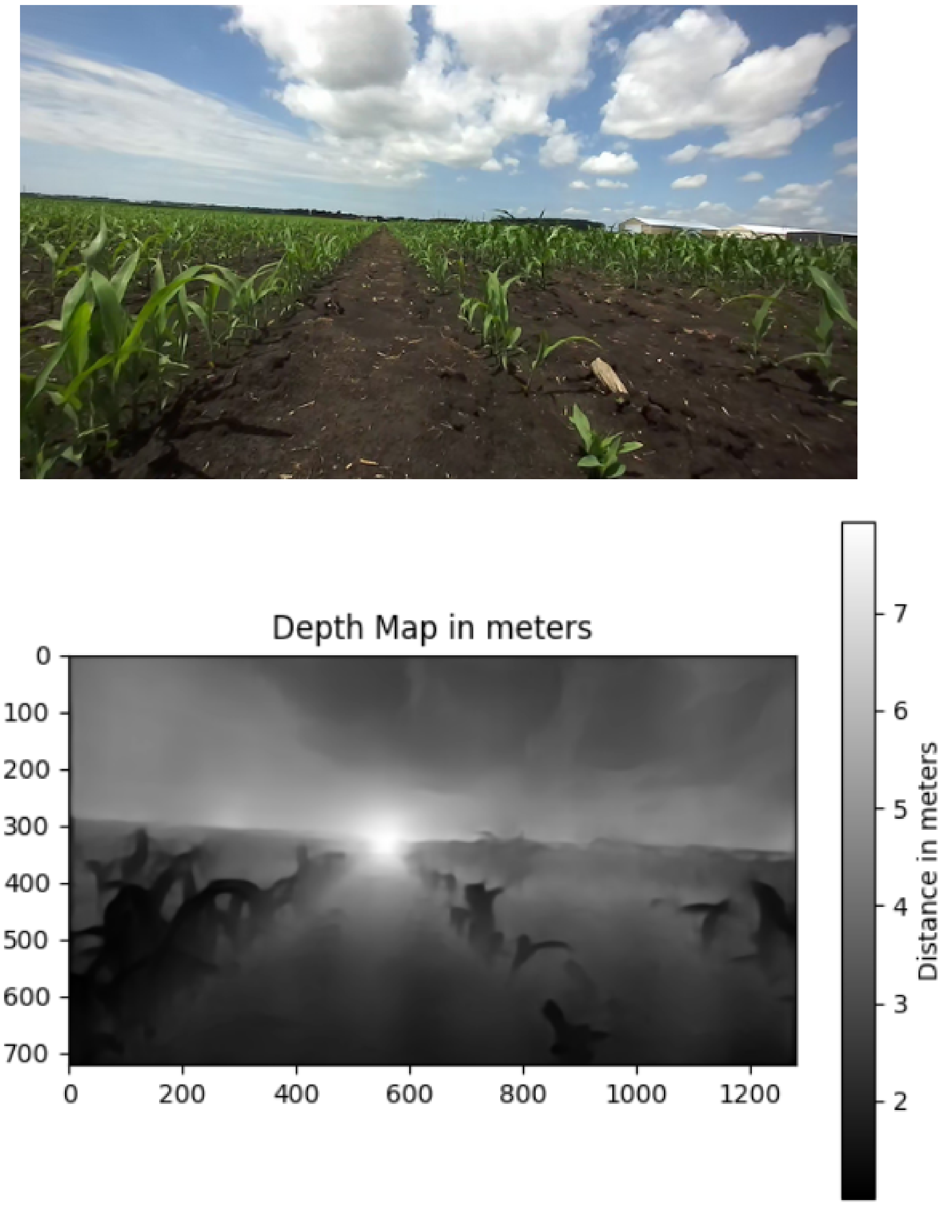

Figure 4.12: Example of performance of DINOv2 and ZoeDepth

f_1 is trained on D_1 with Mean Squared Error loss applied pixel-wise:

$$\mathcal{L}_1 = \frac{1}{wh} \sum_{x=1}^{w} \sum_{y=1}^{h} (d_{pseudo(x,y)} \cdot m_{sky(x,y)} - f_1(i_{raw})_{(x,y)})^2$$

w and h are the width and height of the image. The training gives us the weights of the decoder ($w_{decoder}$) and first depth head (w_{head1}).

- By fixing the weights of f_{dino} and $f_{decoder}$, we can construct the second model:

$$f_2 = \{\bar{f}_{dino}, \bar{f}_{decoder}, f_{head2}\}$$

  The model f_2 is trained on D_2 with binary cross-entropy loss applied pixel-wise:

$$\mathcal{L}_2 = -\frac{1}{wh} \sum_{x=1}^{w} \sum_{y=1}^{h} \left[m_{sky(x,y)} \log(f_2(i_{raw})_{(x,y)}) + (1 - m_{sky(x,y)}) \log(1 - f_2(i_{raw})_{(x,y)}) \right]$$

  The training gives us the weights of the second depth head (w_{head2}).

- For inference on a single RGB image I, $\bar{f}_{dino}$ and $\bar{f}_{decoder}$ are first used to generate a vector embedding $\mathbf{e}_{decoder}$. $\mathbf{e}_{decoder}$ is then fed into $\bar{f}_{head1}$ and $\bar{f}_{head2}$ simultaneously, generating a estimated metric depth $\tilde{d}$ and a mask for the sky m_{sky}. The final estimated metric depth map d can be determined by:

$$\tilde{d}(m_{sky} == 0) = \infty$$
$$d = \tilde{d}$$

A more detailed illustration of the model can be found in Figure 4.13.

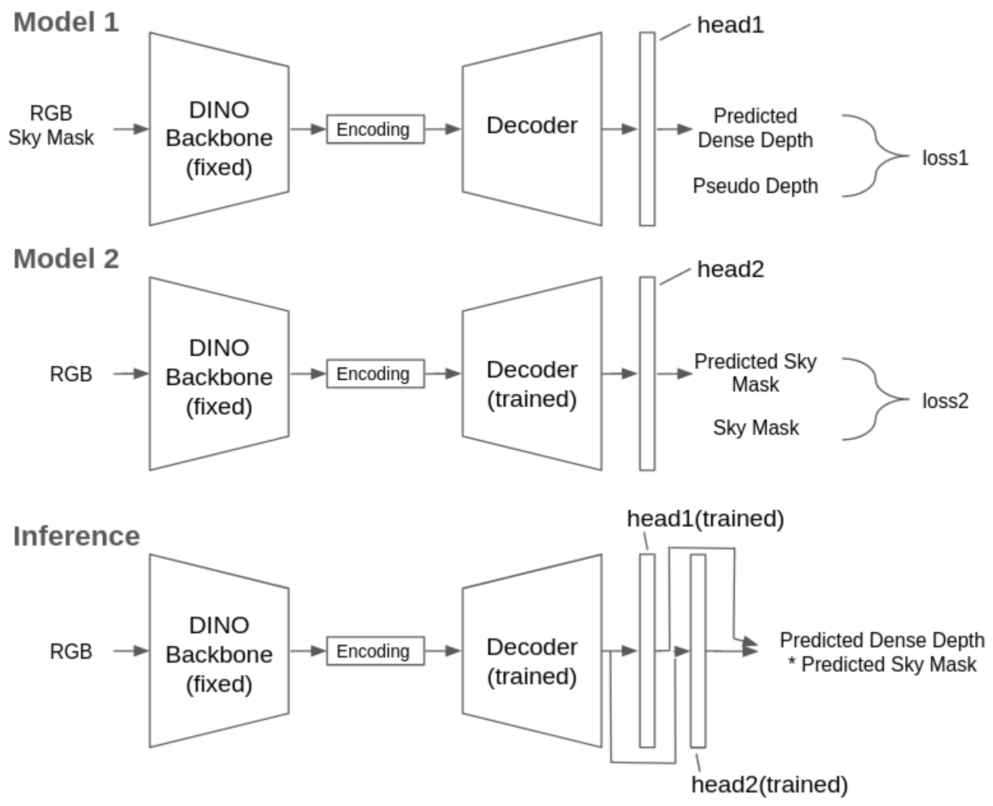

Figure 4.13: Depth Estimation Model Structure

5 RESULTS

5.1 Dataset

To train the monocular depth estimation model, we use a total of 5855 frames collected in agricultural fields, from the dataset [22], of which 1298 are for under-canopy scenes. Figure 4.2 is an example of under-canopy scenes, while Figure 4.12 is an example of above-canopy scenes. We retrieve the RGB image, sparse depth map, and camera pose from the dataset. Applying the method described in Section 4.2.1, pseudo ground-truth depth maps and masks for sky classification can be generated. The camera pose is used in the dynamic view synthesis model.

5.2 Performance of Monocular Depth Estimation

The training and test results of f_1 and f_2 are show in Table 5.1. Combining both models to generate metric depth map predictions results in an MSE against the psuedo ground-truth labels of around 0.003, measured on the test split. Some predicted depth maps are shown in Figure 5.1.

We also run the model on 1000 random frames with size 700×392 included in the dataset, and find the average inference time is around 43 ms per frame using a RTX 2080 GPU.

Table 5.1: Training and test results of monocular depth estimation models

	f_1	f_2
Training Loss	0.000245	0.022228
Test Loss	0.000282	0.022253

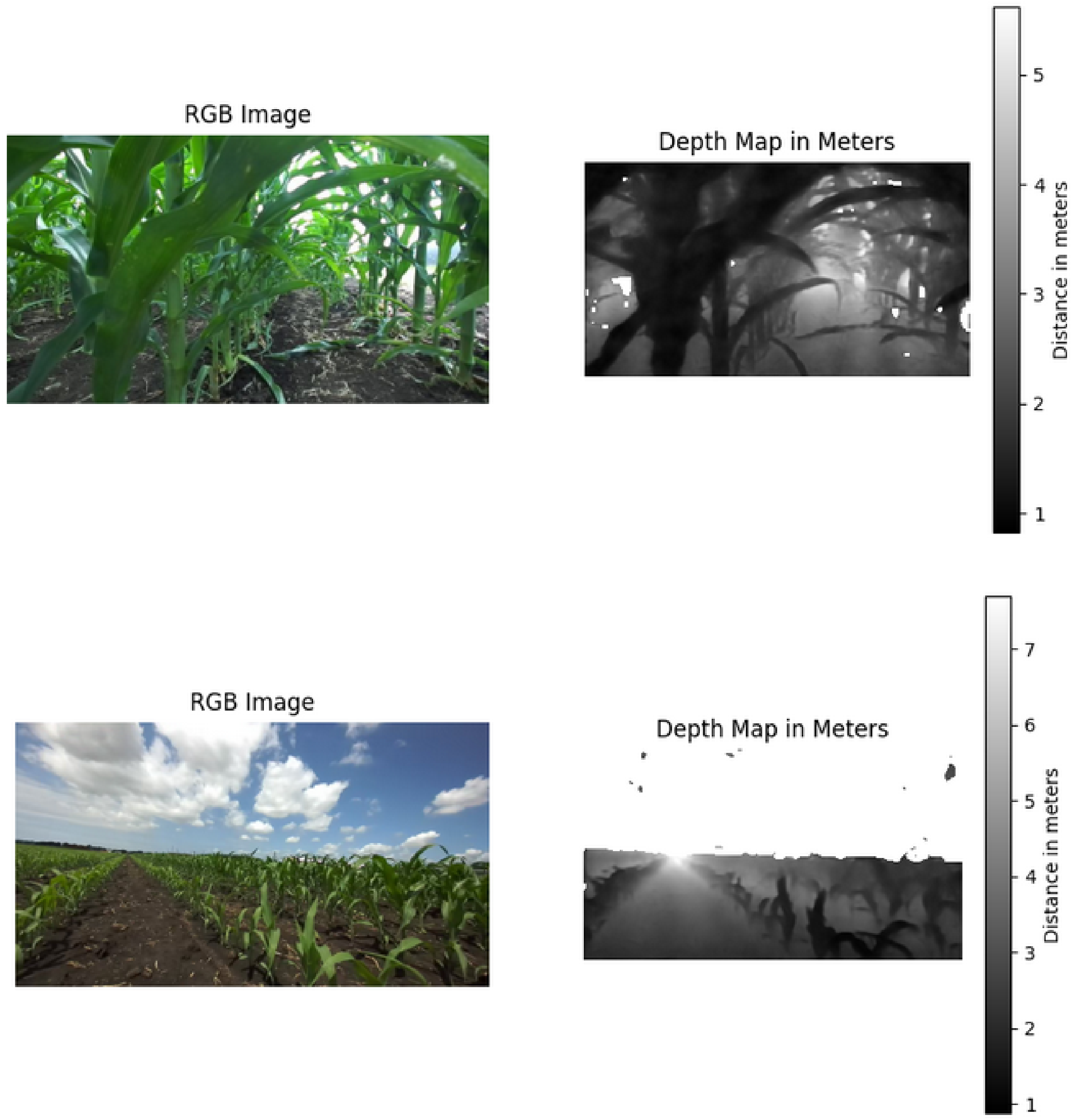

Figure 5.1: Example of depth maps produced by the monocular depth estimation model

5.3 Performance of Dynamic View Synthesis

By combining the monocular depth estimation model and dynamic view synthesis model, we evaluate the final result produced by the overall architecture. Similar to the metrics presented in the work [2], we choose to evaluate the model by using RMSE, PSNR, SSIM, and LPIPS between the ground-truth frame f_{n+1} and predicted frame $\hat{f}_{n+1}$.

- **RMSE (Root Mean Square Error)**: The root mean squared error evaluated on pixel-wise between two images
- **PSNR (Peak Signal-to-Noise Ratio)**: An expression for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects its fidelity
- **SSIM (Structural Similarity Index Measure)**: A method for measuring the similarity between two images, focusing on structural information, luminance, and contrast
- **LPIPS (Learned Perceptual Image Patch Similarity)**: A metric that uses deep neural networks to assess the perceptual similarity between two images, reflecting more accurately human visual perception compared to traditional metrics

Table 5.2 shows the average result after examining 1000 frames with and without the monocular depth estimation model respectively. The result proves that more accurate depth maps with the monocular depth estimation model can greatly improve the result. Figure 5.2 and 5.3 provide the intuitive comparison.

The result is worse compared with the one given in the work [2], as shown in 5.3. However, the authors make several unrealistic assumptions during evaluation that result in an unfair advantage. Firstly, they rely on ground-truth depth map collected in the simulator. Our estimated depth map cannot reach such accuracy as the depth sensor data can be incredibly noisy. Secondly, the dataset they use does not contain many scenes from field environments, which implies that object motions are mostly not random. Nevertheless, in our case, random motions of objects, such as leaves of corn stalks, can severely affect the estimation of optical flow that represents object motions.

Table 5.2: Quantitative results of the dyanmic view synthesis model

	RMSE↓	**PSNR↑**	**SSIM↑**	**LPIPS↓**
With M.D.E	40.3618	16.021	0.4799	0.375
Without M.D.E	94.0811	8.6872	0.1909	0.8357

Table 5.3: Reported metrics from DeCOMPnet

RMSE↓	**PSNR↑**	**SSIM↑**	**LPIPS↓**
20.68	30.60	0.9314	0.0634

Regarding the timing anaylsis, experiments show that it takes on average about 2.3 seconds to predict a frame with size 700×392 using a RTX 2080 GPU.

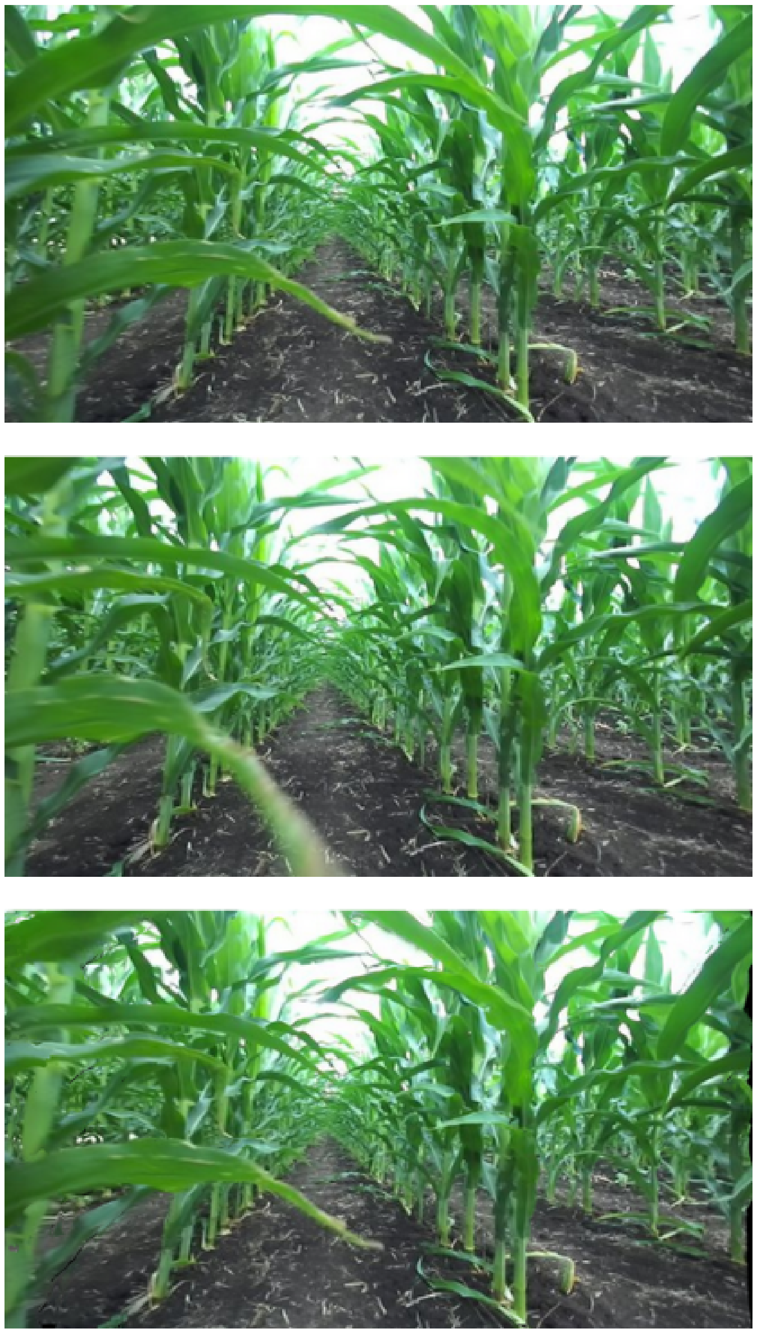

Figure 5.2: Example result from the dynamic synthesis model with dense depth estimation. From top to bottom, frame f_n, ground-truth frame f_{n+1}, and predicted frame $\hat{f}_{n+1}$ respectively

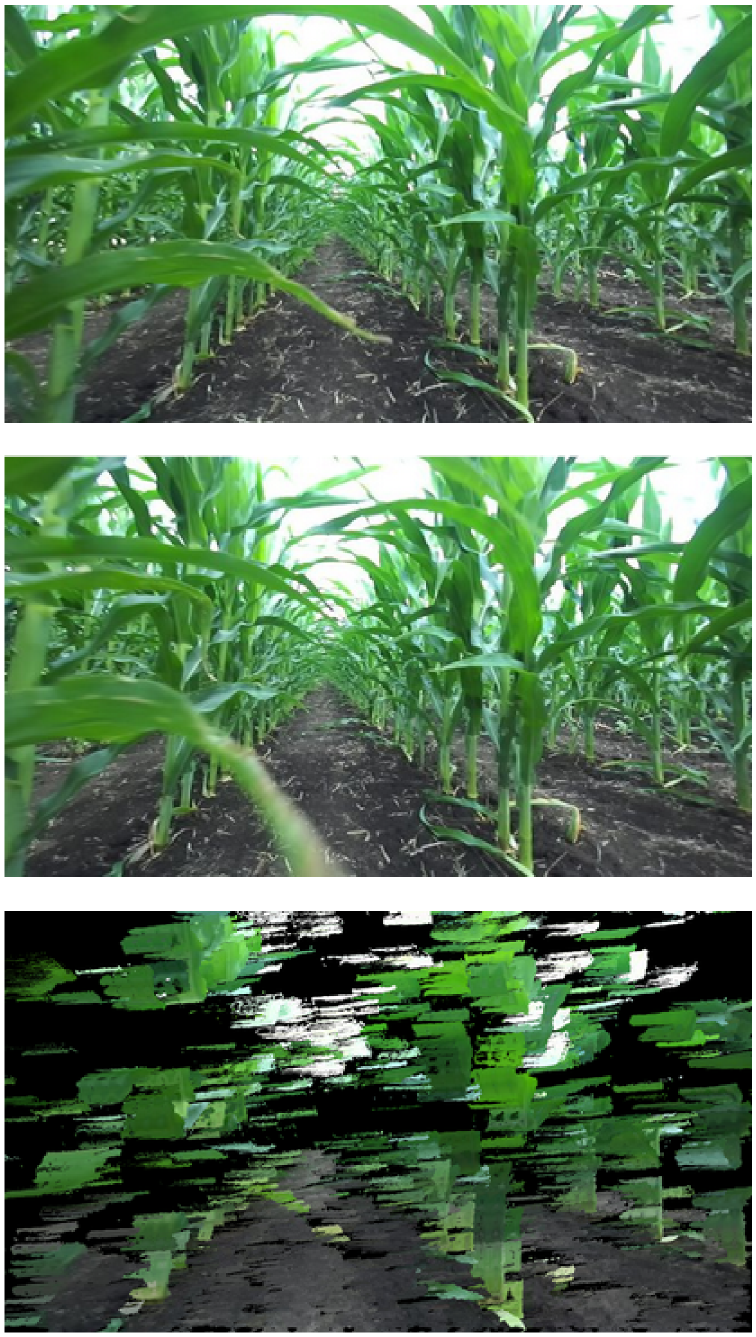

Figure 5.3: Example result from the dynamic synthesis model without dense depth estimation. From top to bottom, frame f_n, ground-truth frame f_{n+1}, and predicted frame $\hat{f}_{n+1}$ respectively

6 CONCLUSION AND FUTURE WORK

In this thesis, we have embarked on an exploration of enhancing teleoperation experiences through advanced video processing techniques, culminating in the proposal of a dynamic view synthesis model. This model is uniquely designed to operate within complex field environments, aiming to significantly improve the fluidity of video sequences by increasing their frame rate. A pivotal component of our approach is the integration of an efficient monocular depth estimation model, designed to overcome the prevalent challenge of obtaining accurate depth maps from onboard cameras—a common limitation in teleoperated systems.

Our work has led us to develop a system that not only promises to enhance the quality of teleoperation video feeds but also opens the door to further advancements in real-time application scenarios. However, the path forward requires a meticulous investigation into the model's feasibility for real-time execution on field robots. The inference time (2.3 seconds) to run the complete model on a single frame is not optimal, especially when the typical delay for video sequences is limited to 100 ms. This entails a series of rigorous experiments designed to evaluate the model's performance and efficiency in live operational contexts. Further investigations on model structures and complexity are also needed.

One of the significant barriers in validating and refining the depth estimation component of our model is the inherent difficulty in validating its accuracy without access to ground-truth depth maps, especially in complex field environments. This challenge necessitates the development of innovative evaluation methods that can reliably assess the performance of the monocular depth estimation model under these constraints. Crafting such methods will be essential for advancing our understanding and capabilities in depth perception within unstructured environments, thereby enhancing the model's overall effectiveness and reliability.

Looking ahead, the potential applications of our dynamic view synthesis model extend beyond immediate improvements in video frame rate for teleoperation. By incorporating further designs to predict future camera poses instead of given ground-truth ones, we can further expand the model's utility to video predictions. This capability can revolutionize how teleoperated systems interact with their environment, enabling more seamless and intuitive control by predicting and adjusting to future states. Such advancements hold the promise of significantly reducing the cognitive load on operators and improving the operational efficiency and safety of teleoperated systems.

REFERENCES

[1] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson, “Synsin: End-to-end view synthesis from a single image,” 2020.

[2] N. Somraj, P. Sancheti, and R. Soundararajan, “Temporal view synthesis of dynamic scenes through 3d object motion estimation with multi-plane images,” in *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, Oct. 2022. [Online]. Available: http://dx.doi.org/10.1109/ISMAR55827.2022.00100

[3] M. Oliu, J. Selva, and S. Escalera, “Folded recurrent neural networks for future video prediction,” 2018.

[4] J.-T. Hsieh, B. Liu, D.-A. Huang, L. Fei-Fei, and J. C. Niebles, “Learning to decompose and disentangle representations for video prediction,” 2018.

[5] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee, “Learning to generate long-term future via hierarchical prediction,” 2018.

[6] A. K. Akan, E. Erdem, A. Erdem, and F. Güney, “Slamp: Stochastic latent appearance and motion prediction,” 2021.

[7] R. Mahjourian, M. Wicke, and A. Angelova, “Geometry-based next frame prediction from monocular video,” 2017.

[8] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely, “Pushing the boundaries of view extrapolation with multiplane images,” 2019.

[9] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, “Nerf: Representing scenes as neural radiance fields for view synthesis,” 2020.

[10] J.-H. Cho, W. Song, H. Choi, and T. Kim, “Hole filling method for depth image based rendering based on boundary decision,” *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 329–333, 2017.

[11] J. S. Yoon, K. Kim, O. Gallo, H. S. Park, and J. Kautz, “Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera,” 2020.

[12] C. Gao, A. Saraf, J. Kopf, and J.-B. Huang, “Dynamic view synthesis from dynamic monocular video,” 2021.

[13] V. Kanchana, N. Somraj, S. Yadwad, and R. Soundararajan, “Revealing disocclusions in temporal view synthesis through infilling vector prediction,” 2021.

[14] S. Farooq Bhat, I. Alhashim, and P. Wonka, “Adabins: Depth estimation using adaptive bins,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, June 2021. [Online]. Available: http://dx.doi.org/10.1109/CVPR46437.2021.00400

[15] S. F. Bhat, I. Alhashim, and P. Wonka, “Localbins: Improving depth estimation by learning local distributions,” 2022.

[16] J.-H. Lee and C.-S. Kim, “Monocular depth estimation using relative depth maps,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9721–9730.

[17] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, “Zoedepth: Zero-shot transfer by combining relative and metric depth,” 2023.

[18] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, “Dinov2: Learning robust visual features without supervision,” 2024.

[19] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging properties in self-supervised vision transformers,” 2021.

[20] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, “Stereo magnification: Learning view synthesis using multiplane images,” 2018.

[21] R. Ranftl, A. Bochkovskiy, and V. Koltun, “Vision transformers for dense prediction,” 2021.

[22] J. Cuaran, A. E. B. Velasquez, M. V. Gasparino, N. K. Uppalapati, A. N. Sivakumar, J. Wasserman, M. Huzaifa, S. Adve, and G. Chowdhary, “Under-canopy dataset for advancing simultaneous localization and mapping in agricultural robotics,” *The International Journal of Robotics Research*, vol. 0, no. 0, p. 02783649231215372, 0. [Online]. Available: https://doi.org/10.1177/02783649231215372