

Iterative Motion Compensation for Canonical 3D Reconstruction from UAV Plant Images Captured in Windy Conditions

Andre Rochow^{*123}, Jonas Marcic^{*1}, Svetlana Seliunina¹, and Sven Behnke¹²³

Abstract—Three-dimensional (3D) phenotyping of plants plays a crucial role for understanding plant growth, yield prediction, and disease control. We present a pipeline capable of generating high-quality 3D reconstructions of individual agricultural plants. To acquire data, a small commercially available Unmanned aerial vehicle (UAV) captures images of a selected plant. Apart from placing ArUco markers, the entire image acquisition process is fully autonomous, controlled by a self-developed Android application running on the drone’s controller. The reconstruction task is particularly challenging due to environmental wind and downwash of the UAV. Our proposed pipeline supports the integration of arbitrary state-of-the-art 3D reconstruction methods. To mitigate errors caused by leaf motion during image capture, we use an iterative method that gradually adjusts the input images through deformation. Motion is estimated using optical flow between the original input images and intermediate 3D reconstructions rendered from the corresponding viewpoints. This alignment gradually reduces scene motion, resulting in a canonical representation. After a few iterations, our pipeline improves the reconstruction of state-of-the-art methods and enables the extraction of high-resolution 3D meshes. We provide the source code of our reconstruction pipeline and a dataset consisting of multiple plants from various crops, captured across different points in time⁴.

Index Terms—Aerial Systems; Applications, Field Robots, Robotics and Automation in Agriculture and Forestry.

I. INTRODUCTION

IN agriculture, phenotyping individual plants is essential for detecting pests, diseases, and assessing growth. 3D reconstruction provides plant scientists with a powerful tool to study plants in greater detail (see Fig. 1). Unmanned ground vehicles (UGVs) have been used to capture multiple images of a plant simultaneously [1]. However, such systems are typically expensive, and need driving access to the plants of interest. In this work, we focus on data acquisition using commercially available and cost-effective drones, such as the DJI Mini Pro 3. Unmanned aerial vehicles (UAVs) can approach a plant anywhere on the field and can be used with plants of any size. By adjusting the flight path based on the plant size, we can monitor the plant growth at any stage with



a) COLMAP Reconstruction b) Textured Mesh (Ours)

Fig. 1: a): Aligned (dense) scene reconstruction using COLMAP [8], including the estimated camera poses. b): Textured mesh extracted after 100 iterations with our proposed method (3D Gaussian Splatting [5] was used as the baseline).

the single UAV. Accurately reconstructing plants in 3D using UAV imagery is particularly challenging due to downwash generated by the UAV, which causes substantial leaf motion. Compared to UGV solutions [1], images can no longer be captured simultaneously from multiple perspectives.

Several methods based on Neural Radiance Fields (NeRF) [2] have been proposed to handle dynamic scenes, including Non-Rigid NeRF [3] and Nerfies [4]. More recently, 3D Gaussian Splatting [5] has emerged as a new state-of-the-art approach, effectively replacing NeRF in many 3D reconstruction tasks. Following its success, prior concepts from deformable NeRFs have been adapted to 3D Gaussian Splatting to handle dynamic scenes [6], [7]. These methods typically model the complete motion within a scene, enabling interpolation not only across viewpoints but also over time. In contrast, our goal is canonical 3D reconstruction for plant phenotyping. Therefore, we model motion solely for the purpose of compensating for it.

We present an approach that can be combined with any 3D reconstruction method and iteratively aligns the input images into a canonical configuration using optical flow (see Fig. 4). Motion is compensated by selectively deforming the original input images. In the first iteration, the raw images are used directly. In subsequent iterations, deformations of the original images are computed by rendering the intermediate 3D reconstruction from the input viewpoints and estimating the optical flow between these renderings and the original inputs. This flow is then used to warp the input images, progressively reducing scene motion. We demonstrate that our method significantly improves the performance of various 3D reconstruction algorithms in the presence of motion.

We automate the image capture process to eliminate the tedious and time-consuming task of manual data collection,

Manuscript received: October, 6, 2025; Revised January, 27, 2026; Accepted February, 26, 2026.

This paper was recommended for publication by Editor Loianno, Giuseppe upon evaluation of the Associate Editor and Reviewers’ comments.

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy–EXC 2070–390732324.

*Equal contribution. Contact: rochow@ais.uni-bonn.de

¹Autonomous Intelligent Systems, University of Bonn, Germany

²Center for Robotics, University of Bonn, Germany

³Lamarr Institute for Machine Learning and AI, Germany

Digital Object Identifier (DOI): see top of this page.

⁴Code & Dataset: <https://github.com/AIS-Bonn/PhenoIMC3D>

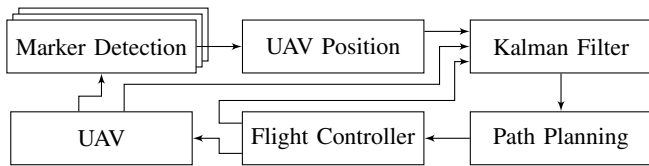


Fig. 2: Pipeline of the autonomous capturing method. The components are described in Section III.

thus reducing the need for human involvement. We tested our approach by capturing plants of different sizes, from 5 cm to almost 2 m high. The primary challenge here is localization from the limited amount of sensor data provided by the UAV. We therefore used fiducial markers placed around the plant to be captured. The UAV’s state is estimated using an extended Kalman Filter. The objective was to reach an adjustable set of waypoints and collect the visual data. Marker information can later be used to enforce the correct scale of the COLMAP [8] reconstruction.

Our contributions can be summarized as follows:

- (i) Autonomous image capturing pipeline for commercially available UAVs.
- (ii) Dataset containing images of different types of plants at different stages of growth, including the camera parameters.
- (iii) A 3D reconstruction pipeline that builds on arbitrary 3D reconstruction baseline methods and iteratively removes motion from the scene via optical flow compensation.
- (iv) A detailed evaluation that highlights that our method greatly improves the results of the baseline methods.

II. RELATED WORK

a) UAV Localization and Flight Control: UAV localization and path following are necessary parts of many applications and were, therefore, extensively researched over the years. The approaches used differ significantly for indoor and outdoor scenarios. Most of the outdoor applications rely on global navigation satellite systems (GNSS), such as the global positioning system (GPS), combined with an inertial navigation system (INS) inside a sensor fusion framework for pose estimation [9], [10]. Other methods also combine GNSS data with other relative positioning systems [11]. The idea is that the relative position or INS provides short-term accurate data, but will drift in the long term. On the other hand, GNSS does not suffer from error accumulation, but has a big error margin and provides data at a lower rate. 4

Often radio communications reception issues and interferences make GNSS unreliable. One possible modality to mitigate this issue is vision. For outdoor localization, absolute visual localization, which involves matching UAV visual data with reference data, is commonly used [12]. In the absence of reference data, one has to rely on relative visual localization using visual-inertial odometry (VIO) and simultaneous localization and mapping (SLAM) [13].

Visual localization is less effective in environments with low visual feature density or repetitive features. One way of addressing these issues is to use fiducial markers [14]. They

are commonly used for indoor localization [15], [16], [17] and for identifying specific places in an outdoor environment, for example, landing zones [18].

b) 3D Reconstruction: Several 3D reconstruction methods that learn a neural radiance field [2] for reconstructing a scene from image inputs [19]–[22]. Müller *et al.* [19] propose to use a hash-based embedding to improve optimization accuracy and speed. Based on this Rosu and Behnke [20] replace the voxel-based hash encoding with a permutohedral lattice that allows for faster optimization in higher dimensions. Instead of sampling densities along a ray, a signed distance function (SDF) is optimized, which significantly improves the quality of the mesh extracted from the volume. Marks *et al.* [23] propose a predefined template-based approach for mesh extraction of plants from UAV imagery, whereas our method operates directly on the observed data and does not require any prior template.

More recently, 3D Gaussian Splatting approaches have been proposed that do not require neural networks to represent scenes. Instead, the scene is modeled as a set of Gaussian primitives, each described by position, orientation, opacity, and shape [5]. Extensions of this approach include methods that model surfaces instead of volumes, allowing for more accurate mesh extraction [24], [25]. Instead of volumetric 3D Gaussians Huang *et al.* [24] use flat 2D Gaussians and Yu *et al.* [25] make use of additionally learned opacity fields. All of these methods assume static scenes. Violations of this assumption lead to blurry reconstructions.

To model motion in non-rigid scenes, a number of NeRF-based methods [2] have been proposed [3], [4], [6]. These methods introduce an additional neural network to estimate deformation of a canonical volume over time [3], [4], [6].

Similarly, dynamic variants of 3D Gaussian Splatting [5] employ deformation networks that apply per-Gaussian offsets, rotations, and scalings to model temporal changes [7], [26]. Zhu *et al.* [27] extends this approach by incorporating optical flow to guide the deformations.

While these methods support interpolation in time, they are not explicitly optimized to produce a sharp canonical representation of the scene. In contrast, our approach does not aim to model deformations but reconstruct a static volume from images containing motion. We propose to iteratively compensate motion by deforming input images into a motion-free representation before reconstruction. Iterative motion compensation has been explored in domains such as medical imaging [28] and video reconstruction [29]. It relies on gated acquisitions or temporally ordered data. Our method operates on unordered UAV images without motion signals.

III. AUTONOMOUS CAPTURING METHOD

Fig. 2 gives an overview of our autonomous image capture. We will discuss each component in the following.

A. UAV

For our task, we used the DJI Mini 3 Pro UAV. All localization and navigation code was executed on the DJI RC Pro remote controller. We developed a custom Android

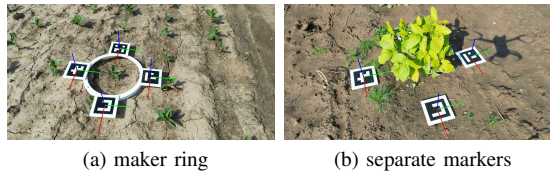


Fig. 3: Marker sets with examples of detected markers.

application that established the communication between the UAV and the controller via the DJI Mobile SDK. Visual data, UAV orientation, altitude, velocity, and gimbal orientation were received and utilized for localization. In return, our controller sent velocity commands to the UAV. GPS data without corrections from another device decreased the accuracy in position estimation, so we chose not to use it.

B. Marker Detection

We selected 4×4 ArUco markers and utilized the OpenCV library [30] to detect them and estimate the camera pose relative to the center. We produced two sets of markers: the first set has smaller markers attached to a ring, while the second set consists of larger, separate markers for bigger plants, as shown in Fig. 3.

In the future, we plan to use a visual inertial odometry pipeline, which will mitigate the need for placing and detecting markers around each plant. It will allow us to place markers only at the corners of the field and then navigate to individual plants using the visual inertial odometry. This will significantly improve the autonomy of our method and speed up the capturing process.

C. UAV Position

The mean position of the markers represents the world coordinate system around which the UAV will navigate. For the ring, we know how the markers are located relative to the center (${}^wT_{0..3}$). For separate markers, we measure the distance from the first marker to the center and fix wT_0 . The transformations between other markers and the center are estimated on the fly:

$${}^wT_{1..3} = {}^wT_0 ({}^cT_0)^{-1} {}^cT_{1..3}, \quad (1)$$

where ${}^cT_{0..3}$ are current marker positions relative to the camera. To estimate the UAV's pose, we first position the markers so that the plant is centered. After obtaining the marker poses relative to the camera ${}^cT_{0..3}$, we can use the transformation between a marker and the center ${}^wT_{0..3}$ to estimate the camera pose relative to the center wT_c . Finally, by adjusting the orientation by the gimbal tilt, we determine the UAV pose wT_r .

D. Kalman Filter

To achieve consistent camera pose estimation and address marker occlusions, we utilize an extended Kalman filter. In addition to pose from visual localization, we combine altitude, attitude, and velocity data during the prediction step. The update step utilizes the velocity command sent to the UAV.

E. Path Planning

To capture the plant from various angles, we fly multiple circles at different heights around the plant. As we increase the height, the gimbal tilt increases, and the radius of the circles decreases accordingly. By adjusting the number of circles and waypoints, height, and tilt, we can capture high-resolution visual data from all desired angles. These parameters are set before the flight and are chosen based on the height and width of the plant.

F. Flight Controller

The goal of motion planning in photo mode is to reach the specified waypoint, defined by its position and orientation. At each step, we calculate the velocity using a feed-forward plus a proportional-integral feedback controller with integral windup and imposed maximal velocity limitation.

$$\dot{\theta}(t) = K_f \dot{\theta}_d(t) + K_p \theta_e(t) + K_i \int_0^t \theta_e(t) dt \quad (2)$$

with $\theta_e(t) = \theta_d(t) - \theta(t)$, where K_f, K_p, K_i are forward, proportional and integral coefficients respectively, $\theta(t)$ - actual trajectory, $\theta_d(t)$ - desired trajectory.

Setpoint ramping is utilized to adjust the waypoint position gradually, preventing abrupt changes when a new waypoint is set. The camera's orientation at each step is adjusted to focus on the center.

The performance of the controller depends heavily on the weather conditions. Depending on the wind, autonomously capturing a single circle with 25 waypoints took five to ten minutes, with an average of 25 min for the entire plant (four circles). The capturing process can be significantly sped up by collecting videos instead of photos, which allows us to complete one circle in under a minute. However, the maximum image resolution will be 4K instead of 8K. We will extend our work to video capturing in the future.

G. Scene Alignment

We use fiducial markers to deduce the correct scale of the scene, which can be done even if they are visible on a small subset of images. To do so, we extract camera positions from images with markers and use them to align the sparse model from COLMAP [8]. The distance between markers and the center of the plant is either known for the marker ring or estimated from the mean camera poses for separate markers.

H. Dataset

The dataset that we publish consists of high-resolution images of several crop types collected at different stages of their growth. They are grouped by scenes that consist of many images collected at different attitudes, distances to the plant, and camera tilt angles. Most of the time, we flew four circles around the plant with a height increase between the second and third ones and tilt angles of 40, 50, 50, and 60 degrees, respectively. Dataset content is further described in Table I. Most scenes consist of 100 images with 25 images per circle,

but sometimes the number of pictures per scene varies due to an incomplete flight or the plant not being in focus.

Additionally, we provide the aligned sparse scene reconstruction from COLMAP with extracted camera poses, which can be used in a dense reconstruction pipeline. We also make available additional scenes with manually collected images, as described in Table II.

Plant Type	Soybean		Soybean		Sugar		Total	
	Corn	(Green)	(Yellow)	Beet	Bean	Beet		Kale
Plants	5	3	2	7	4	1	1	23
Scenes	16	17	14	36	5	1	2	91
Images	1574	1699	1421	3570	480	100	200	9044

TABLE I: Dataset structure. The scenes were autonomously captured in the growing season 2024.

Plant Type	Soybean Sugar		Soybean		Soybean		Total		
	(Yellow)	Beet	Radish	Potato	Kale	(Green)		Corn	
Scenes	8	13	2	5	6	12	5	9	60
Images	1218	2234	173	830	567	1806	933	948	8709

TABLE II: Scenes that were manually captured in the growing season 2023.

IV. 3D CANONICAL PLANT RECONSTRUCTION

Our full 3D reconstruction pipeline, which is designed to align a non-rigid scene in a canonical representation, is illustrated in Fig. 4 and explained in the following Sections IV-A to IV-C. In agricultural fields, gusts of wind can cause plants to move. Furthermore, the UAV that sequentially captures images of the plant produces a significant amount of downwash as well. This leads to non-corresponding leaf poses in the captured images. Our objective is therefore to estimate a canonical (motion-free) 3D reconstruction from a non-rigid scene with unpredictable leaf motion. Input to our method is a UAV-captured scene consisting of images and optimized camera parameters.

A. Train 3D Gaussian Splatting (Step 1)

We train 3D Gaussian Splatting [5] with the original images captured by the UAV. 3D Gaussian Splatting represents the scene using a large number of Gaussians. These primitives are initialized with the sparse point cloud generated by COLMAP. Each Gaussian is described with its position (mean), covariances (orientation and form), opacity, and color via spherical harmonics. These parameters are optimized through differentiable rendering. Furthermore, 3D Gaussian Splatting [5] can add or remove Gaussians to achieve an optimal visual appearance.

Given a set of n images $I^t = \{I_1^t, I_2^t, \dots, I_n^t\}$ at time step t , corresponding camera information $C = \{C_1, C_2, \dots, C_n\}$ and a Gaussian Trainer \hat{T} , we therefore compute a Gaussian scene representation θ_{gs}^t with:

$$\theta_{gs}^t = \hat{T}(I^t, C). \quad (3)$$

$I^{t=0}$ are initialized with ground truth images $I^{t=0} = I^{gt}$.

Note that our pipeline can also be used with neural rendering methods.

B. Render Gaussian Scene from Input Views (Step 2)

The 3D Gaussians get projected by a tile-based rasterizer in order to compute 2D output images. A visibility-aware α -blending sorts all primitives to ensure correct depth layering [5]. For a Gaussian Splatting Renderer \hat{R} , we render a set of predicted images \hat{I}^t with the same camera information C we used for training such that:

$$\hat{I}^t = \hat{R}(\theta_{gs}^t, C). \quad (4)$$

This yields a predicted image \hat{I}_k^t for each ground truth image I_k^{gt} , such that \hat{I}_k^t closely resembles I_k^{gt} .

C. Estimate Optical Flow and Deform Images (Step 3)

We estimate optical flow from rendered images to corresponding ground truth images using RAFT [31]. RAFT takes a pair of images I_1 and I_2 and computes a dense displacement field that maps each pixel in I_1 to a corresponding location in I_2 . For every ground truth image I_k^{gt} and its corresponding prediction image \hat{I}_k^t , we compute a dense displacement field

$$f_k = \mathcal{F}(\hat{I}_k^t, I_k^{gt}), \quad (5)$$

where \mathcal{F} downsamples the images, estimates the optical flow from I_k^{gt} to \hat{I}_k^t , and finally upsamples the resulting flow to match the input image size. Now we apply the displacement field f_k to ground truth image I_k^{gt} to deform it into the predicted image \hat{I}_k^t . For each pixel (u, v) of I_k^{gt} , we get a new current deformed image I_k^{t+1} , with

$$I_k^{t+1}(u, v) = I_k^{gt}((u, v) + f_k(u, v)). \quad (6)$$

D. Iterative Optimization

The steps described in Sections IV-A to IV-C are repeated for M iterations using the updated image set I^{t+1} as input for next iteration. Note that camera parameters C are unchanged.

This iterative refinement regime compensates motion in the scene and gradually aligns the input images in a motion-free canonical representation. This results in sharper predictions and significantly better 3D reconstructions.

E. Mesh Extraction

We crop a region of interest for mesh extraction of θ_{gs} . This is done by removing every primitive that is not inside a radius of r to the center, where r is chosen manually according to the size of the plant. With cropped scene $\tilde{\theta}_{gs}$, we now render a set of cropped images \tilde{I} and optimize a 2D Gaussian volume [24] with them:

$$\tilde{I} = \tilde{R}(\tilde{\theta}_{gs}, C) \quad (7)$$

$$\theta_{2Dgs} = \tilde{T}(\tilde{I}, C) \quad (8)$$

In contrast to 3D Gaussian Splatting methods, 2D Gaussian Splatting [24] uses flat 2D Gaussian disks which are placed directly on the object surface. Because 2D Gaussian Splatting is explicitly modeling the surface of objects, it is better suited to extract a mesh.

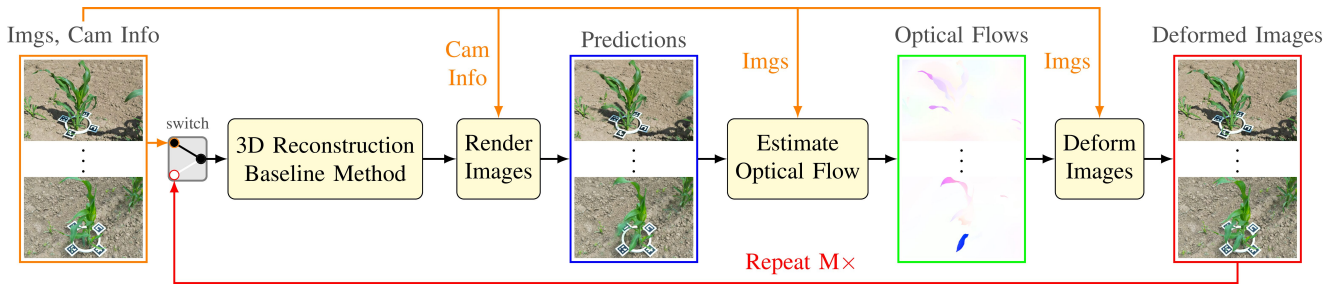


Fig. 4: Canonical 3D Reconstruction Pipeline. In the first iteration, the original input images are used to perform 3D reconstruction. Subsequently, the input views are then rendered with the same camera parameters (Predictions). Using the predictions and original images, we then estimate the optical flow from the predictions to the original input images, which is then used to deform the input images into the current predictions (Deformed Images). In the following iterations, reconstruction is performed using the deformed images instead. These steps are repeated for a predefined number of iterations.

The mesh extraction is done as proposed by Huang *et al.* [24] with a Marching Cubes voxelgrid resolution of 1536^3 . However, we optimize the texture in a separate step. We merge close vertices, decimate vertices to 500-700k [32] and apply HC Laplacian Smoothing [33] using MeshLab [34]. We then extract a UV texture map in 8k resolution from the mesh with Blender [35] Smart UV project using a angle limit of 25 degrees and a margin island of 0.

We then optimize the texture UV map using a differentiable rendering approach. We draw a cropped image \tilde{I}_k and its corresponding camera information C_k . We then render an image \bar{I}_k from mesh \hat{M} with camera parameter C_k . Finally, we compute the L_1 loss between \bar{I}_k and \tilde{I}_k , backpropagate the loss to the texture map and optimize it. This yields a high-resolution texture map.

V. EXPERIMENTS

To evaluate our reconstruction pipeline, we conduct an ablation study and compare our method against other 3D reconstruction baseline methods. We demonstrate that iterative optical flow compensation improves the quality of 3D Gaussian Splatting methods for our objective.

Note that the computation time scales linearly with the number of iterations. The optical flow estimation and image deformation take approximately four minutes per iteration. Memory usage and inference time remain the same as for the specific baseline method.

Due to scene motion, 3D Gaussian Splatting [5] tends to place Gaussian primitives near the camera. To avoid this kind of overfitting, we remove all Gaussian primitives with a z-value less than 30cm within a radius of 60cm around the camera during optimization.

All reported experiments are conducted with an image resolution of 2016×1512 .

In the following, we refer to 3D Gaussian Splatting [5] as (GS) and to Deformable 3D Gaussian Splatting [7] as (DGS). Furthermore, GS+Ours-X or DGS+Ours-X refer to the respective baseline method combined with X iterations of our proposed optical flow compensation.

A. Qualitative Results

1) *3D Gaussian Splatting*: In order to investigate the benefits of our pipeline using methods that are not explicitly modeling motion in scenes, we compare against standard 3D Gaussian Splatting proposed by Kerbl *et al.* [5].

We compare with 3D Gaussian Splatting, proposed by [5], as the baseline method. In our experiments, we applied 100 iterations of our optical flow compensation step, requiring approximately four days on an NVIDIA A6000 GPU.

In Fig. 5, the effectiveness of our motion-compensating optical flow procedure becomes clearly visible. Across all tested scenes, we observe a consistent improvement in visual quality. The most significant improvement occurs between the baseline 3D Gaussian Splatting results and our method with 30 iterations. The improvements from 30 to 100 iterations are smaller compared to the first iterations. However, there are still refinements (see green spy box of Plant 4).

2) *Deformable 3D Gaussian Splatting*: Our approach is also compatible with other 3D reconstruction methods, such as Deformable 3D Gaussian Splatting proposed by Yang *et al.* [7]. They address non-static scenes by explicitly modeling motion by a deformation network. During the optimization process, the deformation network learns to adjust the Gaussian primitives in response to the observed motion, allowing for a more accurate representation of dynamic scenes.

To integrate DGS into our pipeline, we introduce a few modifications. First, we apply the removal of Gaussian primitives located too close to the camera, as described in Section V. In addition, we define the first image as the canonical frame and apply our optical flow compensation only to the remaining images in the dataset, warping them into the corresponding reconstructions obtained with the time embedding of the canonical frame.

For the optical flow compensation within Deformable 3D Gaussian Splatting [7], we limit the process to ten iterations, because the deformation network already captures a significant portion of the motion. Further iterations are not necessary and would significantly increase computation time. Note that ten iterations already require approximately four days on a NVIDIA A6000 GPU.

As the results of Fig. 6 show, our optical flow compensation improves the visual quality of baseline Deformable 3D Gaus-

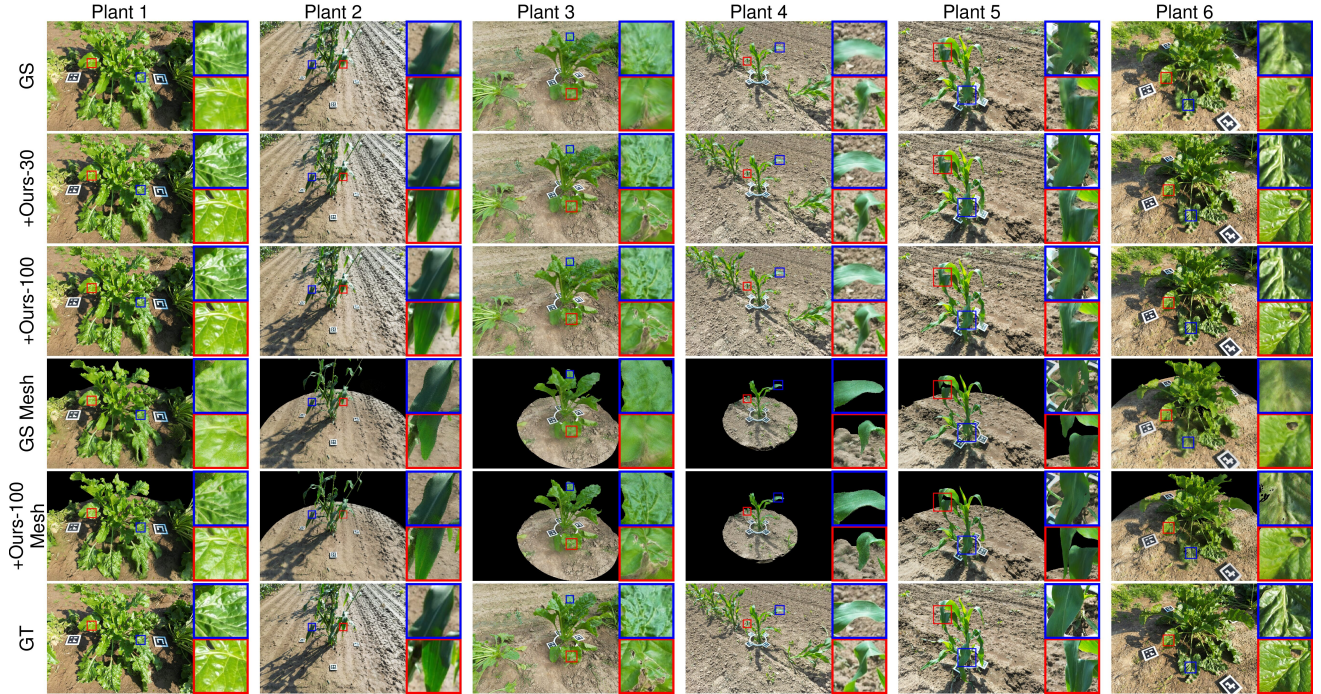


Fig. 5: Comparison of 3D Gaussian Splatting [5] (GS) and our proposed method with optical flow compensation. The first row (GS) shows results from standard 3D Gaussian Splatting without motion compensation. The second and third rows (+Ours-30 and +Ours-100) show our results after 30 and 100 iterations of optical flow compensation, respectively. The fourth and fifth rows (GS Mesh and +Ours-100 Mesh) show meshes extracted from the 2D Gaussian representation before and after compensation. The last row (GT) presents the ground truth images for reference.

sian Splatting [7] in the reconstructed scenes. As expected, these improvements are smaller than in the case of standard 3D Gaussian Splatting [5], which is also reflected in our quantitative evaluation in Table III.

B. Quantitative Results

We present a quantitative evaluation in Table III, using the mean PSNR, LPIPS, SSIM, and FID [36] scores computed across all input images from 11 randomly selected scenes.

The FID score is calculated between the ground truth input images and the reconstructed images. Although these pairs often do not align perfectly due to leaf motion in the scene, a lower FID indicates that the distribution of generated images more closely matches the distribution of real images in the feature space. In contrast, PSNR, SSIM, and LPIPS assume a direct correspondence between the generated and ground truth images. Since our method reconstructs a canonical scene that compensates for leaf motion, direct comparison with the ground truth is not feasible. To address this, we use optical flow to deform the generated images into the viewpoint of

the ground truth, and evaluate the deformed predictions. The accuracy of the optical flow improves when the predicted image closely resembles a spatially deformed version of the ground truth image at the same camera pose. As shown in Table III, our method significantly outperforms the baseline methods across all evaluation metrics. Fig. 7 shows that our method yields the greatest improvements in the initial iterations. This demonstrates that our method already provides significant benefits with fewer iterations and shorter training times when resources and time are limited.

Other Baseline Methods: To show that our proposed framework is highly flexible with respect to the choice of baseline 3D reconstruction methods, we additionally apply it with 30 iterations to Triangle-Splatting+ [37], Permuto-SDF [20], Instant-NGP [19], and Beta-Splatting [38]. As shown in Fig. 9 and Table IV, the reconstruction quality of these baseline methods also improves significantly over the course of the iterative process. Note that these methods were integrated into our framework without modification.

C. Limitations in Mesh Extraction

While our method significantly enhances the texture quality of reconstructed scenes (see Figs. 5 and 6), its impact on geometry is more modest, but still evident (see Fig. 8). Notably, the geometry of stems and leaves that appear blurry or are missing in the baseline method is better recovered when meshes are extracted after applying our proposed optical flow compensation. Since geometry extraction relies on 2D Gaussian Splatting [24], which is inherently limited in

Method	PSNR \uparrow	LPIPS \downarrow	SSIM \uparrow	FID \downarrow
DGS [7]	24.15	0.1670	0.8160	3.8830
+Ours-10	24.31	0.1530	0.8395	3.5613
GS [5]	24.41	0.2060	0.7895	6.0689
+Ours-100	25.37	0.1505	0.8502	3.6547

TABLE III: Quantitative evaluation of our method compared with the two baseline methods DGS [7] and GS [5].

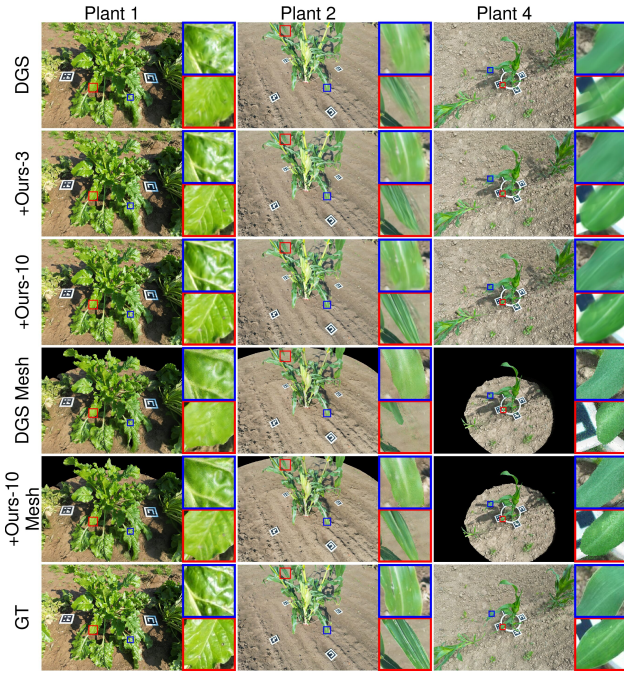


Fig. 6: Comparison of Deformable 3D Gaussian Splatting [7] (DGS) and our proposed method with optical flow compensation. The first row (DGS) shows results from standard Deformable 3D Gaussian Splatting. The second and third rows (+Ours-3 and +Ours-10) show our results after 3 and 10 iterations of optical flow compensation. The fourth and fifth rows (DGS Mesh and +Ours-10 Mesh) show extracted meshes and the final row (GT) shows the ground truth.

representing fine-scale geometry [24], and is further affected by errors introduced by the Marching Cubes algorithm, not all improvements introduced by our method are faithfully reflected in the extracted meshes. For example, a very thin stem accurately recovered by our approach may still be poorly captured during mesh extraction.

D. Failure Cases of Optical Flow Compensation

When the motion is too large, the initial canonical estimate after iteration 0 does not adequately capture the scene content. As a result, the optical flow estimation fails in the absence of reliable correspondences. Fig. 10 presents results of our framework on images from the D-NeRF dataset [6], where it fails to produce a satisfactory canonical estimate.

Method	PSNR \uparrow	LPIPS \downarrow	SSIM \uparrow	FID \downarrow
TS+ [37]	23.16	0.2162	0.7002	11.053
+Ours-30	23.88	0.1713	0.7578	7.7899
BS [38]	24.99	0.2253	0.7935	6.1283
+Ours-30	25.66	0.1628	0.8435	3.5930
I-NGP [19]	22.61	0.4333	0.5589	19.257
+Ours-30	23.66	0.3474	0.6514	11.720
P-SDF [20]	21.00	0.4508	0.5155	21.916
+Ours-30	21.56	0.3646	0.5914	15.105

TABLE IV: Quantitative evaluation of our method using non-Gaussian-Splatting 3D reconstruction baseline methods.

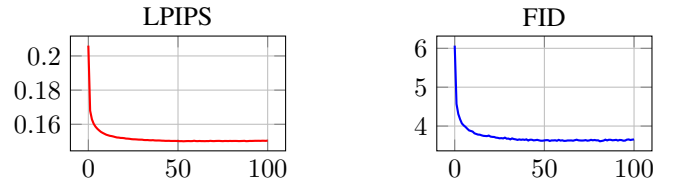


Fig. 7: LPIPS and FID scores (y-axis) of GS+Ours plotted after each iteration (x-axis), following the quantitative evaluation from Table III.

In contrast, plant scenes typically exhibit smaller motion induced by wind, making our method well suited to this setting. Note that there is no accumulation of warping errors, because the optical flow is always estimated from scratch in each iteration.

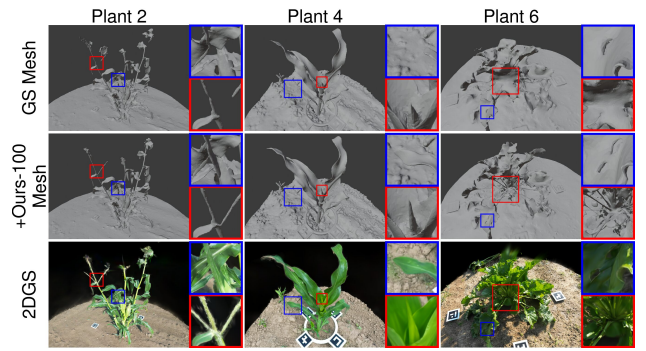


Fig. 8: We compare geometry of GS [5] (first row) and GS combined with our proposed method after 100 iterations (second row). Meshes are extracted using 2D Gaussian Splatting [24] (2DGS) as described in Section IV-E. The third row shows the rendered image of the 2DGS volume before extracting the mesh of GS+Ours-100 (second row). The zoom-in regions highlight improvements in geometry.

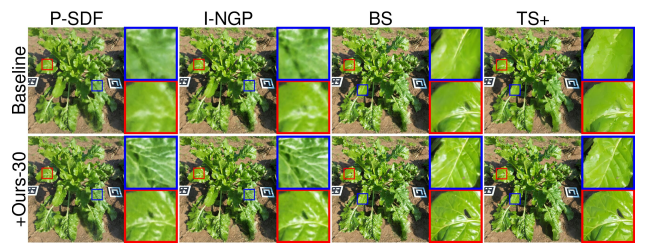


Fig. 9: Qualitative comparison of our framework with non-Gaussian-Splatting 3D reconstruction baselines.

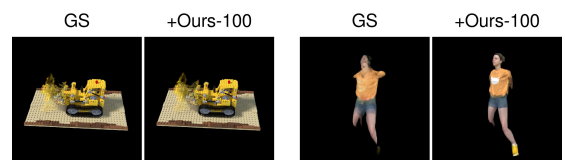


Fig. 10: Failure cases of our proposed optical flow compensation on the D-NeRF dataset [6].

VI. CONCLUSION

We presented a framework that builds on 3D reconstruction baseline methods to create a high-resolution reconstruction from plant images containing a significant amount of motion. Especially the downwash from the UAV creates significant plant motion in the sequentially captured images. Our framework improves the results of both, rigid and non-rigid 3D reconstruction baseline methods, as shown in the experiments. In addition, we release a dataset of agricultural plants collected with a self-developed Android application controlling on a small and affordable UAV. Future work includes improving mesh extraction techniques, refining optical flow estimation, and reducing training time.

REFERENCES

- [1] F. Esser, R. A. Rosu, A. Cornelißen, L. Klingbeil, H. Kuhlmann, and S. Behnke, "Field robot for high-throughput and high-resolution 3D plant phenotyping: Towards efficient and sustainable crop production," *IEEE Robotics & Automation Magazine*, vol. 30, no. 4, pp. 20–29, 2023.
- [2] 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," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.
- [3] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt, "Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 12959–12970.
- [4] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, "Nerfies: Deformable neural radiance fields," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 5865–5874.
- [5] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 4, pp. 139–1, 2023.
- [6] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, "D-NeRF: Neural radiance fields for dynamic scenes," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 10318–10327.
- [7] Z. Yang, X. Gao, W. Zhou, S. Jiao, Y. Zhang, and X. Jin, "Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 20331–20341.
- [8] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [9] A. Nemra and N. Aouf, "Robust INS/GPS sensor fusion for UAV localization using SDRE nonlinear filtering," *IEEE Sensors Journal*, vol. 10, no. 4, pp. 789–798, 2010.
- [10] G. Zhang and L.-T. Hsu, "Intelligent GNSS/INS integrated navigation system for a commercial UAV flight control system," *Aerospace science and technology*, vol. 80, pp. 368–380, 2018.
- [11] G. Mao, S. Drake, and B. D. Anderson, "Design of an extended kalman filter for uav localization," in *Information, Decision and Control*, 2007, pp. 224–229.
- [12] A. Couturier and M. A. Akhloufi, "A review on absolute visual localization for UAV," *Robotics and Autonomous Systems*, vol. 135, p. 103666, 2021.
- [13] N. Gyagenda, J. V. Hatilima, H. Roth, and V. Zhmud, "A review of GNSS-independent UAV navigation techniques," *Robotics and Autonomous Systems*, vol. 152, p. 104069, 2022.
- [14] M. Kalaitzakis, B. Cain, S. Carroll, A. Ambrosi, C. Whitehead, and N. Vitzilaios, "Fiducial markers for pose estimation: Overview, applications and experimental comparison of the ARTag, AprilTag, ArUco and STag markers," *Journal of Intelligent & Robotic Systems*, vol. 101, pp. 1–26, 2021.
- [15] H. Lim and Y. S. Lee, "Real-time single camera SLAM using fiducial markers," in *ICCVS-SICE*, 2009, pp. 177–182.
- [16] G. Zhenglong, F. Qiang, and Q. Quan, "Pose estimation for multi-copters based on monocular vision and AprilTag," in *Chinese Control Conference (CCC)*, 2018, pp. 4717–4722.
- [17] J. Bacik, F. Durovsky, P. Fedor, and D. Perdukova, "Autonomous flying with quadcopter using fuzzy control and ArUco markers," *Intelligent Service Robotics*, vol. 10, pp. 185–194, 2017.
- [18] P. H. Nguyen, K. W. Kim, Y. W. Lee, and K. R. Park, "Remote marker-based tracking for UAV landing using visible-light camera sensor," *Sensors*, vol. 17, no. 9, p. 1987, 2017.
- [19] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–15, 2022.
- [20] R. A. Rosu and S. Behnke, "PermutoSDF: Fast multi-view reconstruction with implicit surfaces using permutohedral lattices," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 8466–8475.
- [21] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "NeRF++: Analyzing and improving neural radiance fields," *arXiv preprint arXiv:2010.07492*, 2020.
- [22] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 5855–5864.
- [23] E. Marks, F. Magistri, and C. Stachniss, "Precise 3d reconstruction of plants from uav imagery combining bundle adjustment and template matching," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 2259–2265.
- [24] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2D Gaussian splatting for geometrically accurate radiance fields," in *ACM SIGGRAPH*, 2024, pp. 1–11.
- [25] Z. Yu, T. Sattler, and A. Geiger, "Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 6, pp. 1–13, 2024.
- [26] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang, "4D Gaussian splatting for real-time dynamic scene rendering," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 20310–20320.
- [27] R. Zhu, Y. Liang, H. Chang, J. Deng, J. Lu, W. Yang, T. Zhang, and Y. Zhang, "Motionings: Exploring explicit motion guidance for deformable 3d gaussian splatting," *Neural Information Processing Systems (NeurIPS)*, vol. 37, pp. 101790–101817, 2024.
- [28] X. Zhu, M. Chan, M. Lustig, K. M. Johnson, and P. E. Larson, "Iterative motion-compensation reconstruction ultra-short te (imoco ute) for high-resolution free-breathing pulmonary mri," *Magnetic resonance in medicine*, vol. 83, no. 4, pp. 1208–1221, 2020.
- [29] X. Yang and C. Yang, "Imrnet: An iterative motion compensation and residual reconstruction network for video compressed sensing," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 2350–2354.
- [30] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
- [31] Z. Teed and J. Deng, "RAFT: Recurrent all-pairs field transforms for optical flow," in *European Conference on Computer Vision (ECCV)*, 2020, pp. 402–419.
- [32] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *Conference on Computer Graphics and Interactive Techniques*, 1997, pp. 209–216.
- [33] J. Vollmer, R. Mencl, and H. Mueller, "Improved Laplacian smoothing of noisy surface meshes," in *Computer Graphics Forum*, Wiley Online Library, vol. 18, 1999, pp. 131–138.
- [34] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, G. Ranzuglia, et al., "Meshlab: An open-source mesh processing tool," in *Eurographics Italian Chapter Conference*, vol. 2008, 2008, pp. 129–136.
- [35] Blender Online Community, *Blender - a 3d modelling and rendering package*, Blender Institute, Amsterdam: Blender Foundation, 2025. [Online]. Available: <https://www.blender.org/>.
- [36] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," *Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.
- [37] J. Held, R. Vandeghen, S. Son, D. Rebain, M. Gadelha, Y. Zhou, M. C. Lin, M. Van Droogenbroeck, and A. Tagliasacchi, "Triangle splatting+: Differentiable rendering with opaque triangles," *arXiv preprint arXiv:2509.25122*, 2025.
- [38] R. Liu, D. Sun, M. Chen, Y. Wang, and A. Feng, "Deformable beta splatting," in *Special Interest Group on Computer Graphics and Interactive Techniques Conference*, 2025, pp. 1–11.