Learned Visual Navigation for Under-Canopy Agricultural Robots

Arun Narenthiran Sivakumar[1] Sahil Modi[2] Mateus Valverde Gasparino[1] Che Ellis[3]
Andres Eduardo Baquero Velasquez[1] Girish Chowdhary*,[1,2] Saurabh Gupta*,[4]
[1]Department of Agricultural and Biological Engineering, University of Illinois at Urbana-Champaign (UIUC)
[2]Department of Computer Science, UIUC, [4]Department of Electrical and Computer Engineering, UIUC,
[3]EarthSense Inc.

***Abstract*—This paper describes a system for visually guided autonomous navigation of under-canopy farm robots. Low-cost under-canopy robots can drive between crop rows under the plant canopy and accomplish tasks that are infeasible for over-the-canopy drones or larger agricultural equipment. However, autonomously navigating them under the canopy presents a number of challenges: unreliable GPS and LiDAR, high cost of sensing, challenging farm terrain, clutter due to leaves and weeds, and large variability in appearance over the season and across crop types. We address these challenges by building a modular system that leverages machine learning for robust and generalizable perception from monocular RGB images from low-cost cameras, and model predictive control for accurate control in challenging terrain. Our system, CropFollow, is able to autonomously drive 485 meters per intervention on average, outperforming a state-of-the-art LiDAR based system (286 meters per intervention) in extensive field testing spanning over 25 km.**

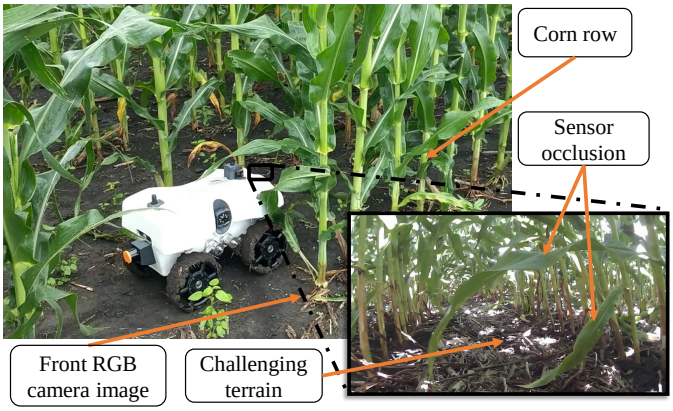

Fig. 1: CropFollow is an autonomous navigation system for under-canopy agriculture robots. It uses RGB images from a front-facing camera to output steering commands to drive the robot in crop rows.

I. Introduction

This paper describes the design of a visually-guided navigation system for compact, low-cost, under-canopy agricultural robots for commodity row-crops (corn, soybean, sugarcane etc), such as that shown in Figure 1. Our system, called CropFollow, uses monocular RGB images from an on-board front-facing camera to steer the robot to autonomously traverse in between crop rows in harsh, visually cluttered, uneven, and variable real-world agricultural fields. Robust and reliable autonomous navigation of such under-canopy robots has the potential to enable a number of practical and scientific applications: High-throughput plant phenotyping [43, 37, 68, 66, 58, 25], ultra-precise pesticide treatments, mechanical weeding [41], plant manipulation [17, 61], and cover crop planting [64, 62]. Such applications are not possible with over-canopy larger tractors and UAVs, and are crucial for increasing agricultural sustainability [55, 22].

Autonomous row-following is a foundational capability for robots that need to navigate between crop rows in agricultural fields. Such robots cannot rely on RTK (Real-Time Kinematic)-GPS [21] based methods which are used for over-the-canopy autonomy (e.g. for drones, tractors, and combine harvesters) because of GPS signal attenuation and multi-path errors. The under-canopy row-following task consists of detecting and following the rows of crop, by determining the distance from the rows and the angle relative to the row, and using this to track specified row-relative pose. In a typical 80 acre land-parcel in row-crops, the rows are about 400 meter long and full of visual clutter. The crop rapidly grows during the growing season, rendering a constantly changing visual environment. Therefore, autonomous navigation of under-canopy robots has remained a challenging and open problem. LiDAR is known to work under the canopy and can return geometric information [32]. However, LiDAR is costly, and it does not capture semantic information. For example, LiDAR cannot directly distinguish whether observed occupancy corresponds to untraversable obstacles (actual crop plant stalk), or traversable obstacles (hanging leaves, weeds, uneven terrain). This fundamentally limits LiDAR based methods from estimating distance and angle from the row, leading to low robustness of autonomy, as reported by low distance-between-interventions [32]. This motivates our use of richer sensing and lower-cost modalities in the form of RGB images.

Using RGB images for under-canopy navigation however has proven to be non-trivial and has become a primary bottleneck for under-canopy robotics. Importance of semantics precludes the use of traditional methods that infer geometry

Project website with data and videos: https://ansivakumar.github.io/learned-visual-navigation/.
Correspondence to {av7,girishc}@illinois.edu.
*Girish Chowdhary and Saurabh Gupta contributed equally and are listed alphabetically.

from monocular RGB image streams [45, 23]. Visual variability during the day and the season limits heuristic based crop-lane detection algorithms, and visual similarity results in positional drift with SLAM algorithms[67]. It is clear therefore, that the high variability and clutter in the agricultural environment necessitates the use of learning. However, the lack of large-scale datasets, the difficulty of collecting field data, lack of a clear reward signal, and the infeasibility of building a simulator for this task, makes it challenging to employ machine learning.

Our contribution in this paper is a field-validated modular vision based crop-row following system to overcome the above challenges. We term this system CropFollow, as it provides the foundational row-following capability to small, low-cost robots. Our system decouples perception and control. The perception system uses monocular RGB image from the on-board camera to estimate row-relative robot pose. It does so by directly estimating the robot's relative heading to the row (measured as the angle the robot makes with the row direction), and robot's placement in row (measured as the ratio of distance from the left row to inter-row separation). These data are fused with inertial measurements using a Bayesian sensor fusion system (Extended Kalman filter (EKF)), and utilized to generate row-following control in terms of desired angle and speed for staying in the center of the row using a nonlinear robust controller (Model Predictive Control (MPC)). The ability to directly predict relative heading and distance from monocular RGB images is one key novelty of our approach, and has key efficiency and robustness benefits: the approach avoids having to first detect the plants (which can be many) [27], or explicitly segmenting the ground from plants (which is highly challenging with more clutter in the environment) [67]. Our presented system is able to successfully traverse crop rows regardless of the crop's growth stage. In field trials of about 25 kilometers, our system required fewer interventions than a LiDAR based system [63](485 meters per intervention *vs.* 286 m), while at the same time cutting down sensing cost by 50×. In offline experiments, we find that the proposed perception models generalize well to new crops. These results clearly establish that our modular visual navigation system enables vision based autonomy for under-canopy field robots.

II. Related Work

Autonomous Navigation in Agricultural Fields. GPS, alone and in combination with IMU and RTK corrections, is commonly used for outdoor navigation for tractors and over-canopy agricultural robots [52, 2, 3, 70, 37, 18, 36]. Under-canopy navigation is concerned with autonomous row-following between the rows of crops. In such under-canopy environments, GPS suffers from significant multipath errors and signal attenuation under the canopy [32], furthermore RTK correction signals aren't always available. As an alternative, LiDAR data along with heuristics based algorithms for row-following have been used for under-canopy and orchard navigation [33, 6, 62, 32, 63]. However, LiDAR is costly, sensitivity to noise, and cannot sense semantic or contextual information.

This has motivated vision-based navigation systems. Past work in vision-based agricultural navigation can be classified into over the canopy [72, 26, 69, 34, 4], under-canopy in orchards [59, 51, 7, 1] and under-canopy in row crops and horticultural crops [67, 27]. Vanishing lines based heuristics was commonly used in these works. In orchards and over-canopy visual navigation setting crop rows are clearly visible, which makes heuristic based line fitting possible. However, these algorithms do not directly apply to under-canopy navigation in commodity crops such as corn and soybean (the focus of this paper) where the row-spacing is much tighter (10× smaller than orchards), there is a high degree of visual clutter, complete and frequent occlusion of the camera by leaves, presence of weeds, crop residue on the ground, and changing visual appearance as the crop grows (see Figure 4 and Figure 7 for examples). Incidentally, corn and soybean acerage is atleast 10× larger than orchards. Recent visual servoing with RGB-D has been used for orchar navigation [1], however this approach will not work in corn-soybean canopies due to visual clutter and small-size of crops earlier in the growing season.

Classical Navigation. Navigation in classical mobile robotics [60, 56] follows a modular approach with perception (simultaneous localization and mapping (SLAM)), path planning relative to generated map, and trajectory tracking control. There are various successful SLAM techniques for this in structured and static environments such as in urban self-driving and indoor navigation. However, geometric reconstruction and localization in deformable and dynamic under-canopy agriculture environments is challenging. Furthermore, geometric approaches equate traversability with free space. While generally true, in off-road field settings this is not true (short weeds are fine to run over, hanging plant leaves can be run into), and necessitates the use of learning. Visual-inertial odometry (VIO) based approaches (*e.g.* [50]) that are common in other outdoor navigation tasks are not useful here without a pre-built map, or GPS waypoints to close the loop and prevent drift (see Figure 8), or navigation in non-straight rows.

Learned Navigation. Researchers have used machine learning for navigation and locomotion in situations where heuristics have failed. Learning has been used in different ways: [28, 73, 65] learn high-level semantic cues and statistical regularities for navigation, [39, 19] use learning to provide robustness to actuation noise for path following, while [24, 54, 47, 53, 8] rely on learning to reduce or eliminate the dependence on expensive sensors for collision-free local navigation. Our work falls into this last category. Our use of learning not only eliminates dependence on LiDAR, but surpasses its performance through better discrimination between traversible and intraversible areas by use of learning on camera images. Research in this last category can be further distinguished based on the policy design and supervision used for training. Given the infeasibility of simulation, challenging terrain, lack of a reliable unsupervised self-supervision signal (as used

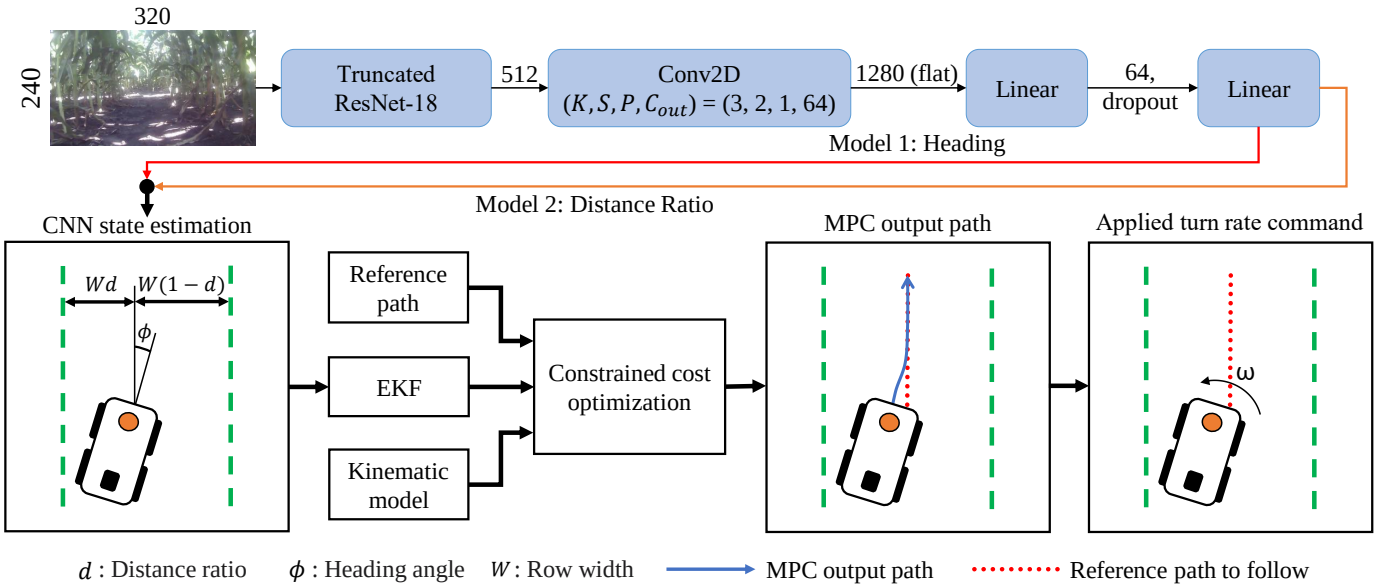

Fig. 2: CropFollow Overview. We use a convolutional network to output robot heading and placement in row. This is used to compute the row center which is used as a reference trajectory. A model predictive controller converts reference trajectories to angular velocity commands.

in BADGR [35]), and difficulty of large-scale field experiments, renders reinforcement learning, imitation learning, and self-supervision based methods infeasible for our task [54, 47, 47, 24, 35, 49, 30]. Also, lack of large-scale datasets for training has prevented the use of machine learning (over-canopy datasets *e.g.* [15, 48], and urban self-driving datasets *e.g.* [10] exist, but aren't useful for under-canopy training). Therefore, we employ a modular approach [5, 44, 11, 42] and use supervised learning for training the perception module. Eliminating trial-and-error from learning improves sample efficiency, and the use of an analytical low-level controller allows easy generalization over varying terrains. Our contribution is in the design and experimental validation of a modular autonomy system in unique, challenging agricultural settings.

Learned Lane Following. Crop row following is similar to lane following in context of self-driving cars, however is much more challenging given no clear lane markings and extreme amounts of clutter. Past lane following works use reactive control based on traditional vanishing line estimation [71]. However, vanishing line estimation is brittle. Consequently, recent works employ learning. [9] trained a vanishing point estimation network from an urban driving dataset with clearly visible lanes. Such lanes are not directly visible in our cluttered under-canopy environments (See Figure 7 and Figure 9). Thus, those models won't work, as is, in our setting. Second, they only output the vanishing point which only tells us about the heading and not the distance ratio (see Appendix Section IV-E). Our method bypasses having to estimate the vanishing point and directly outputs all the necessary information required for the robot to navigate in under-canopy. Therefore, our method is the most direct and efficient way to achieve under-canopy row following. [40, 46, 74] predict semantic segmentation of the scene to estimate lane boundaries, while [13, 57, 14] employ end-to-end learning to directly output control commands via classification or regression). The former techniques require fine-grained pixel level annotations for training, and real-time inference is computationally expensive. End-to-end control is impractical in our setting as mentioned above. [29] learns to predict the location of lane in the image to estimate distance but does not predict heading and distance directly. [12] show that CNNs can be trained to predict driving affordances in uncluttered simulation environments where lane markings are clearly visible. In contrast, our work provides substantial experimental results that demonstrate that CNN based state estimators can lead to high-performing autonomous navigation systems capable of operating in the wild cluttered under-canopy fields, surpassing the current default practice of using a LiDAR.

Closest to our work, Gu *et al.* [27] use learning to detect corn stalks and fit lines. This approach suffers when corn stalks are not visible, and has not been validated in real corn fields. We follow an implicit approach to directly estimate the states (row-relative heading and offset). This allows us to train a machine learning system that is robust to these challenges, as shown by our extensive in-field validation.

III. System Design

Figure 2 shows an overview of our presented system. Images from on-board RGB camera on the robot are processed through a convolutional network to predict robot heading ϕ, and relative placement d between crop rows. This relative placement is converted into the robot's distance from the left and the right crop rows by multiplying with the lane width. These heading and distance predictions are filtered using a Bayesian filter (we use the Extended Kalman Filter) that optionally also fuses them with high-frequency input from an inertial measurement unit. The filtered heading and distances are used to generate a course correcting reference path in the robot coordinate frame. A model predictive controller is used to compute angular velocity commands to achieve this reference path. A lower-

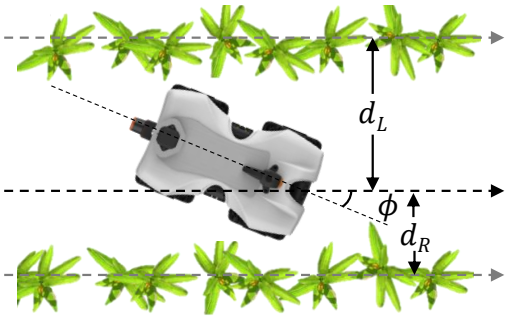

Fig. 3: Our method uses the robot's heading, ϕ and ratio of distance from the left and the right crop row, $d = d_L/(d_L + d_R)$, as the intermediate representation between perception and planning.

level proportional–integral–derivative (PID) controller is used to track the commanded angular velocity.

In this section, we describe the robot platform, the CNN architecture, the Extended Kalman Filter, and the model predictive controller. We describe the data collection and ground truth generation procedure in Section IV.

Robot Platform. TerraSentia is an ultra-compact 4-wheeled skid-steering mobile robot designed to drive through fields and collect data. It has a Raspberry Pi 3 on-board for lower-level motor control and an Intel i7 NUC for data processing and navigation. Note that our unit had no discrete GPU, so the integrated Intel GPU is used for model inference. This robotic system is equipped with various sensors but only 4 are relevant to this paper. There is a dedicated GPS module that determines baseline autonomous driving performance (when GPS signal is reliable). The current LiDAR-based autonomy is fueled by the 2D horizontal-scanning LiDAR (Hokuyu UST-10LX) and a 6 DOF Inertial Measurement Unit (IMU). Finally, our approach utilizes only the forward facing, 720p at 30 fps monocular camera sensor (OV2710) and an IMU. We note specifically that since LiDAR is not utilized in our presented visual system, no *explicit* real-time depth signal is available to the model.

Perception Model. We choose a learning approach due to its superior generalizability compared to color-based segmentation navigation proposed by previous works. Figure 7 shows the classical system's failure to segment the lane in common late stage data. CropFollow's perception model takes in 320×240 RGB images and outputs the robot heading (in degrees) and its relative placement in the crop row. Figure 3 shows how the heading and the relative placement is defined. Heading ϕ is the angle of the robot relative to crop rows. The relative distance d is the ratio of the distance to the left of the row to the lane width, *i.e.* $d = \frac{d_L}{d_L+d_R}$, where d_L and d_R are the distances to left and right crop rows.

The perception model uses a ResNet-18 [31] backbone that has been pretrained on ImageNet [20]. We truncate ResNet-18 right before the average-pooling layer, and add in an additional convolutional layer, a fully connected layer, dropout, and final prediction layer. The final prediction layer outputs the heading ϕ, and the distance ratio d. We found that independent networks to predict heading and distance ratio worked better than a single joint network.

Fig. 4: Sample images from the collected dataset.

IMU Fusion with Extended Kalman Filter. An Extended Kalman Filter [16, 21, 56] was used to reduce the effect of uncertainties in distance and heading estimations by fusing the inertial data with the vision data. We used $s = \begin{pmatrix} d_L & d_R & \phi \end{pmatrix}^T$ as the state. State s_k evolves over time as per the prediction function $f(s_{k-1}, u_{k-1})$ (derived using the robot's kinematics, see supplementary). Here s_{k-1} is the state at the previous time step, and u_{k-1} is the linear and angular velocity at the previous time step. Robot's linear speed v and angular speed ω are calculated from wheel encoders, and IMU respectively. We assume additive zero-mean Gaussian process and measurement noise. As we directly observe s, the measurement function is an identity function. Output from the CNN is used in the update step. More details about the form of the prediction function, and co-variances of the Gaussian noise are provided in the supplementary material.

Model Predictive Controller. We used a non-linear Model Predictive Controller (MPC) to generate angular speed commands to the robot given the reference path to be followed, as shown in Figure 2 [37, 36]. MPC uses the fused output states $s = \begin{pmatrix} d_L & d_R & \phi \end{pmatrix}^T$ from the EKF, the Unicycle kinematic model (see supplementary) of the robot and reference path, which is a straight line through the center of the lane, to solve a constrained optimization problem with the minimum and maximum curvature radius as the constraints. The output is a path defined in terms of the curvature ρ, which determines the angular velocity $\omega = \rho\, v$ where v is the linear velocity. The angular speed for the first point in the output path is applied and the optimization process is repeated. A PID controller is used to maintain the commanded angular speed, based on feedback from IMU's yaw angular speed.

IV. Data Collection and Ground Truthing

Given lack of any under-canopy agriculture datasets, we collected a large dataset by driving the TerraSentia robot under the canopy. We manually operated the robot in 19 corn and 4 soybean fields across Illinois and Indiana, and collected time-series data from the front-facing RGB camera, LiDAR, and IMU. We collected 2.7 hours of corn data and 1.2 hours of soybean data, and made sure to collect data for different growth stages. We also included data where the robot was driven in a zigzag manner. This was done to expose the

perception models to a broader distribution of data that may be experienced during autonomous runs. Figures 4 and 12 shows sample corn and soybean images from the dataset. We note the variability in appearance, occlusion, challenging illumination (shadows, low-light under the canopy), challenging terrain, and leafy plants. This raw data and a subset of annotations will be made available upon acceptance.

Ground Truthing. To train our perception model from Section III, we need labels for robot heading and the ratio of the distance from the left and the right crop rows. Preliminary investigation of using LiDAR for extracting this information for training wasn't fruitful. Hence, we gathered human labels.

However, asking humans for such geometric labels is not easy. Unlike semantic labels, such metric geometric quantities are non-trivial for humans to label. As an example, consider images in Figure 4, and consider speculating the robot heading and placement in the row. To circumvent this issue, we designed an indirect annotation procedure. We asked humans to label the horizon and the vanishing lines corresponding to the crop row (Figure 6 (left)). This together with the camera calibration information allows us to recover the robot heading and placement in row using projective geometry. Figure 5 provides an overview of the different steps involved in computing these quantities from the annotated images. For the case where the horizon is not visible, we instead ask humans to mark out vertical crop stalks (Figure 6 (right)). This allows us to estimate the vanishing point for the vertical direction which readily provides the slope of the horizon. Precise formulae and derivations are provided in the supplementary material.

We annotated a total of $25,296$ corn images. 28% of these are from early growth stage, while 72% are from late growth stage. We split the dataset into a training and a validation set (83% training, 17% validation). We made sure that data from the same video is either entirely in the training set, or entirely in the validation set. Our main experiments use this corn data. We also labeled $10,685$ soybean images (54% early, 46% mid) to study transfer across crops.

V. Experimental Results

Our experiments are designed to test the autonomous crop row traversal capability of our proposed system, effectiveness of the proposed modular policy, and data efficiency and generalization of our learned models. We evaluate these aspects through a combination of offline and online (field) experiments. Offline experiments are conducted on our collected dataset. They allow us to systematically study data efficiency and model generalization, and help us chose models for online experiments. Online experiments are conducted in the field, and allow us to study the interplay between perception and control systems. We also conduct end-to-end evaluation for the task of crop row traversal, and compare against an existing system based on LiDAR [63].

A. Offline Evaluation of Perception Model

Offline evaluation of the perception module is conducted on the collected dataset. All experiments except ones for generalization across crops, use the corn to train and test.

Model	Mean		Median		95%ile	
	ϕ_{err}	d_{err}	ϕ_{err}	d_{err}	ϕ_{err}	d_{err}
Baseline	11.41	0.48	8.81	0.48	30.33	0.65
Combined	2.24	0.08	1.39	0.06	5.37	0.20
Separate	**1.99**	**0.04**	**1.21**	**0.03**	**4.71**	**0.10**

TABLE I: Perception Module Performance: We report L1 error in heading (in $^\circ$) and distance ratio prediction. The trivial baseline model always predicts median ϕ, d from the training set. The combined model learns heading and distance simultaneously, but ultimately performs worse than individually trained models.

Metrics. We measure prediction performance using L1 error in heading and distance ratio predictions, ϕ and d.

Training. We used ResNet-18 [31] pretrained on ImageNet [20] to initialize our models. Models were trained to minimize the $L2$ loss with the Adam optimizer [38] for 50 epochs. We started with an initial learning rate of 10^{-4} and dropped it by a factor of 10 at 40^{th} and 45^{th} epochs. All layers of the network were optimized.

Results. Table I presents the performance of our CNN models. We experimented with 2 variants: predicting heading and distance ratio separately using two models, and a single multi-task network. For reference, we also report the performance for a trivial predictor that always predicts the median heading and distance ratio from the training set. This measures the hardness of the task, puts performance of our model in context.

Both models worked well, with the separate model variant working better. Our best model achieves an average L1 error of 1.99° for heading, and 0.04 for distance ratio. Inference speed for this model on the robot was around 20 FPS, which is fast enough for accurate control (more on this in Section V-D). Our main field experiments are conducted with this model.

B. Comparison with Classical Baselines

Color-based segmentation is a common first step in classical vanishing lines based row following literature. Figure 7 shows the results of automatic color-based segmentation on common late stage data. We see that the segmented lane is not clear. This validates CropFollow's learning-based approach as a general navigation system for all growth stages across the season. To compare with a feature matching based VIO algorithm, Vins-fusion was used as the baseline [50]. To compare with stereo based Vins-fusion as well, data collected from Intel Realsense D435i camera was used only for this experiment and recommended intrinsic values from Realsense library was used as Vins-fusion parameters. Figure 8 demonstrates the heading and cross track error of CropFollow, Vins-fusion with monocular RGB camera and with stereo IR camera. Note that in case of distance the plot shows cross track error (offset distance from the middle of the lane) and not the relative error with respect to the ground truth. The ground truth was calculated by annotating vanishing lines and horizon (same approach as training labels for CropFollow). Ground truth heading and distance ratio at first frame was used to initialize Vins-fusion localization (both monocular RGB and stereo IR). CropFollow is vastly superior to Vins-fusion in distance

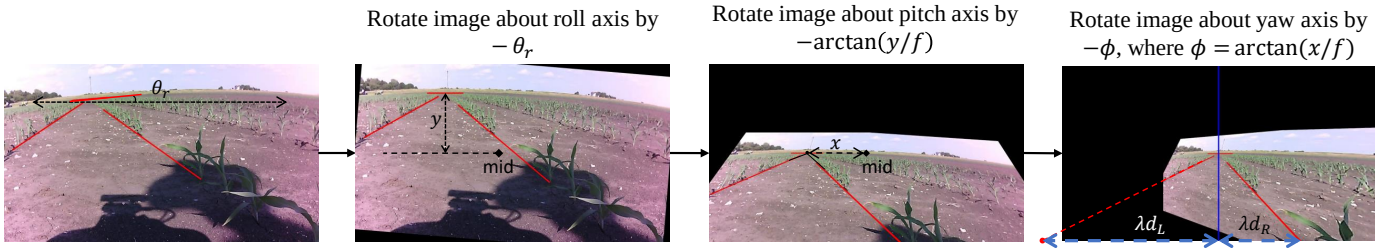

Fig. 5: Ground truthing procedure. Using the horizon annotations, we correct for the camera roll, and pitch. After this, heading, ϕ can be calculated by looking at the crop row vanishing point, and distance ratio can be computed from the intercepts of the crop row lines in the heading corrected image, as $d_L/(d_L + d_R)$.

Fig. 6: Annotations. We annotate the horizon and crop rows for early season images (left). For late season images when the horizon is not visible, we annotate the vertical corn stalks (right).

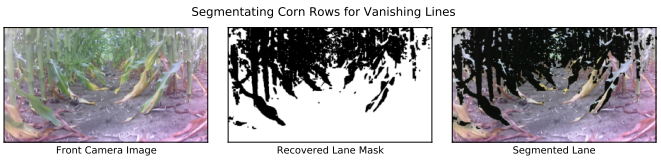

Fig. 7: Classical color-based vanishing line segmentation on late stage data according to related works [7, 67]. We see that crop segmentation does not produce a clear visual of the lane, so automatic vanishing line based lane-following is not possible. In particular, extraneous leaves artificially alter the boundaries of the lane.

prediction as seen from very similar cross track error as ground truth and is comparable in heading. Although Vins-fusion shows comparable heading tracking, it suffers significantly from position drift which is orders of magnitude greater than the lane width between crop rows (about 0.75m) making it impractical to use for row following. This is because there is no opportunity for loop closure in long crop rows. This validates reactive navigation as pursued in CropFollow is a valid approach for row following.

C. In Field End-to-End System Evaluation

We conducted end-to-end system evaluation with the model described above. We compared the performance of the following 2 systems, along with 2 variants each:

- **CropFollow (w/ IMU)**. This is our proposed system that uses the above CNN model for heading and distance ratio prediction, EKF for fusing IMU information, and MPC for executing control commands. We also compare with a variant that does not use IMU information (denoted by CropFollow (w/o IMU)).
- **LiDAR System [63] (w/ IMU)**. This system uses readings from the LiDAR mounted on top of the robot to estimate the robot heading and distance from the crop rows using line fitting. Other parts of the system are same as our system: Use of an EKF to fuse information from the IMU, and use of MPC for generating control commands. We also compare to a variant that does not use IMU information (denoted by LiDAR System [63] (w/o IMU)).

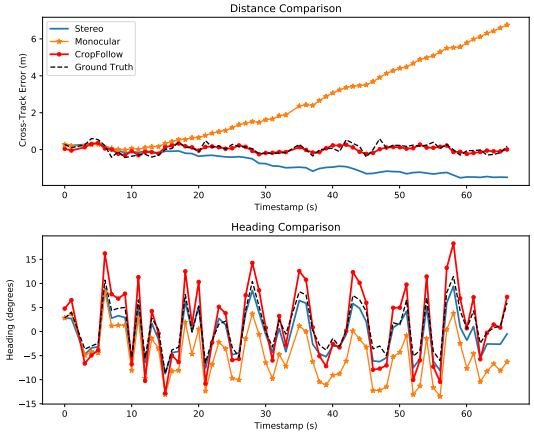

Fig. 8: CropFollow *vs.* Vins-fusion mono *vs.* Vins-fusion stereo. We compare the cross track error (CTE) (offset distance of the robot from the middle of the lane) and heading of CropFollow, Vins-fusion with mono and Vision-fusion with stereo IR at different frames in a trajectory. CropFollow shows better CTE than Vins-fusion.

Fig. 9: Sample images from field trials. Bottom row consists of traditionally adverse conditions for vision-based navigation.

Evaluation Methodology. All 4 systems are tested on the same unique 4.85 km. These 4.85 km come from 15 different experiments that were done in different parts of the field, over different growth stages, different days, different time of the day, and weather conditions. While there is a lot of variability in these 4.85 km, we attempted to minimized the variability in conditions for the 4 systems to ensure result comparability. Runs for the different systems for each of the 15 experiments were done one after another over the same

Growth Stage	Length (in m)	LiDAR w/ IMU	LiDAR w/o IMU	CropFollow w/ IMU	CropFollow w/o IMU
Early	1120	-	-	3	4
Late	3726	13	72	7	8

TABLE II: Field Experiments: We report the number of interventions for the different methods. LiDAR can't operate in early season as crops are too short. Our system can work under both conditions and requires interventions.

Method	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
LiDAR	9	17	8	7	19
CropFollow	0	0	0	0	0

TABLE III: We report the number of interventions of LiDAR w/o IMU and CropFollow w/o IMU by repeating the test on the same row 5 times in the field. CropFollow outperformed LiDAR in all trials. Variation in LiDAR counts shows its sensitivity to noise.

routes, and with the same constant linear robot velocity of 0.6 m/s. Run order for the different systems was randomized to prevent environmental bias. This experiment thus presents results pooled over field trials of 19.4 km. For each method, we measure the number of human interventions needed to complete the experiment. Human interventions were required when the robot crashed into the corn stalks. This metric measures autonomy effectiveness.

Results. Table II reports the number of interventions for the 4 systems that we evaluated. We separately report results for early and late season experiments. Note that LiDAR system from [63] can't operate in early season data since early season corn stalks are shorter than the robot, and not detected by the 2-D LiDAR. Our vision based systems works reasonably well. In late season when the LiDAR based system does work, we note that it had more interventions than our system, 72 *vs.* 8 without IMU, and 13 *vs.* 7 with IMU. Thus, our presented vision-based system outperforms the LiDAR based system, while also reducing sensing cost by 50× ($30 for RGB camera, while $1500 for LiDAR). Note that these are paired experiments done over long run lengths (4.85 km), and the performance gap is statistically significant (with p-value $< 10^{-3}$). To further compare the without IMU versions of the LiDAR system and CropFollow, we did an experiment where LiDAR failed and CropFollow succeeded, and did 4 additional runs for each method (Table III). We found CropFollow to work better than the LiDAR system in all 5 trials. The quality of our output is further shown by the fact that our system is closing the loop only at about 20Hz, *vs.* 40Hz for the LiDAR system, but still achieves a better end performance.

D. Training Data Efficiency and Generalization

The above experiments demonstrate that our proposed system works. We next conduct experiments to measure data efficiency and generalization ability of our trained models. We investigate three questions: How much labeled data did we actually need to get good prediction and field performance? How much data do we need for the next crop? And what is the best use of annotation budget? We answer these questions through a combination of field and offline experiments.

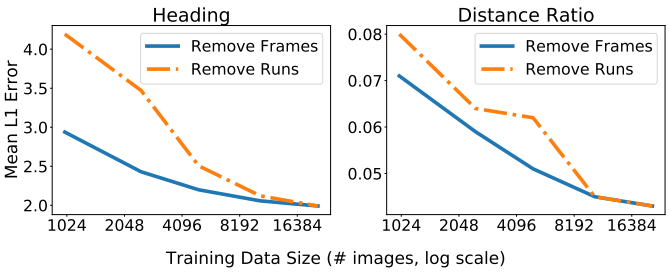

Fig. 10: Performance as a function of amount of training data. We sub-sample training data by either removing entire data collection runs, or by removing frames. Our perception model starts doing well even with small amounts of data.

Fig. 11: Performance when training and testing on early *vs.* late *vs.* combined data. Models trained on only early or only late data don't generalize well, and training on combined data works best.

Data Efficiency for Corn. We first measure the data efficiency of learning through offline experiments. We report the validation performance as a function of the amount of training data. We consider 2 versions obtained by sub-sampling a) at the level of data collection runs; b) at the level of frames. Figure 10 plots performance as a function of training dataset size. We make two observations. First, models start performing well at around $10K$ labeled images. Second, it is more beneficial to label images from many different runs, than many images from a few runs.

We also study if we need data from all growth stages to learn a good model. Figure 11 reports validation performance on each growth stage for models trained on 6000 images of

Number of Training Images	100	1000	20986
Validation Metrics (Mean L1 Error)			
Heading Error	6.28	4.19	1.99
Distance Error	0.09	0.08	0.04
In field Metrics (Number of Interventions)			
CropFollow (w/ IMU) @ 22 FPS	0	0	0
CropFollow (w/ IMU) @ 10 FPS	0	0	0
CropFollow (w/ IMU) @ 5 FPS	4	0	0
CropFollow (w/ IMU) @ 2.3 FPS	failed	0	0
CropFollow (w/o IMU) @ 22 FPS	0	1	0
CropFollow (w/o IMU) @ 10 FPS	1	2	0
CropFollow (w/o IMU) @ 5 FPS	2	0	0
CropFollow (w/o IMU) @ 2.3 FPS	failed	8	9

TABLE IV: Field and offline validation of models trained with 100, 1000 and 20986 images to study training data efficiency.

Fig. 12: Sample early and mid stage soybean. Note the stark difference to corn (right). Soybean is stouter with broader leaves.

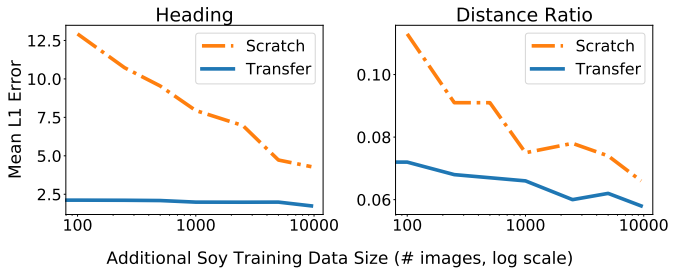

Fig. 13: Generalization from corn to soybean. Model trained on corn (*Transfer*) generalizes well to soybean in comparison to training from ImageNet initialization (*Scratch*).

either early stage, late stage, or an equal combination of both. We note that both models trained on a single growth stage have poor performance on the other growth stage. Our model that is trained on a blend of early and late stage data is most accurate throughout the entire season with an average error of $1.28°$ and 0.03 for heading and distance ratio respectively.

Field Experiments for Data Efficiency for Corn. However, note this is only performance of the perception module in isolation. It will be more instructive to look at the field performance of the whole system as a function of the training set size. Table IV reports field performance of 3 models trained with 100, 1000 and $21K$ images (we took the models that sub-sampled data at the level of runs as they had a sharper drop in performance), in the same crop row of length 428m. Interestingly, we note that at the base control frequency of 20Hz, systems trained with as little as 100 images worked without interventions! It should be noted that this does not mean that 100 images are sufficient for robust and repeatable performance, but shows that the system learns quite a bit with little data, and the modular approach which leverages the IMU and a robust controller is capable of tolerating a less perfect perception system. Indeed, difference in performance is more evident at lower update rates. Perception models trained on larger datasets are likely more robust to extreme viewpoints and hence can recover better from off-center locations that may arise at lower update rates. These results provide information on allowable heading and distance ratio prediction error at different speeds and update rates with which amount of training data needed for training in new crops can be determined. It can be seen that with higher prediction errors, using IMU and higher model update rate makes the system robust.

Generalization to Another Crop. We also study the data efficiency for enabling autonomous navigation for a new crop. We do this via offline experiments and measure how much additional training data is needed to adapt a model trained on corn to achieve good performance. Figure 13 plots the validation metrics as a function of number of Soybean training images (*Glycine max*, Figure 12), for our transferred model, and for a baseline model that starts from ImageNet initialization. We note strong transfer of the model trained on Corn. Even without any training on Soybean, our two models achieve good performance with a average error of just $2.20°$ and 0.07 for heading and distance. Although only from Corn to Soybean, this is a very desirable result. It suggests that our Corn model might already work in Soybean rows with minimal additional labeling. We leave field trials to future work.

E. Error Modes and Stress Testing

To understand the common error modes in our CropFollow and LiDAR system, we visualized the front camera video stream before failures in field experiments. Also, to stress test the proposed CropFollow, field tests were conducted in a challenging field with a sharp curve, occlusions and gaps and experiments were also conducted to test the performance of CropFollow at increased speeds.

Visualization of different error modes. Figure 14 shows the different error modes in CropFollow and LiDAR navigation system. Large gaps in crop rows was the common cause of failure in CropFollow (our training data did not include such cases). Sensor occlusion and bumpy terrain were the other rare causes of failures. In contrast, failure due to gaps was rarely observed in LiDAR since it was specifically engineered to be robust to it. But because of its high sensitivity to noise, even minor sensor occlusion by leaves affects LiDAR performance and leads to interventions. CropFollow's performance in gaps could be improved with adding training data whereas LiDAR's occlusion problem is a sensor limitation.

Stress testing. To test the performance in challenging conditions, CropFollow (w/ and w/o IMU) was tested in a field with sharp curves, gaps and occlusion from weeds. 3 and 6 interventions w/ and w/o IMU respectively was observed in a test of 600m. Last row in Figure 9 shows the challenging condition in this field. Also, CropFollow's performance at higher speeds was tested. CropFollow showed same stable behavior at $1m/s$ but oscillations in trajectory due to latency was observed at $1.4m/s$ or more.

VI. Conclusion

We presented a vision based autonomous under-canopy navigation system. Through a modular architecture and a learning-based approach we showed that machine vision can be applied for reliable and robust navigation in cluttered, changing, and harsh under-canopy environments. 25 km of real-world validation on an under-canopy robot demonstrated that our visual navigation approach is not only $50\times$ more cost-effective than LiDAR but also leads to fewer interventions. Our system forms a new benchmark for visual navigation under the canopy, and our openly accessible dataset (1030 labeled images and 24266 unlabeled images of our corn data) will enable further research. We hope our results and dataset pave the way for wider adoption of learned visual

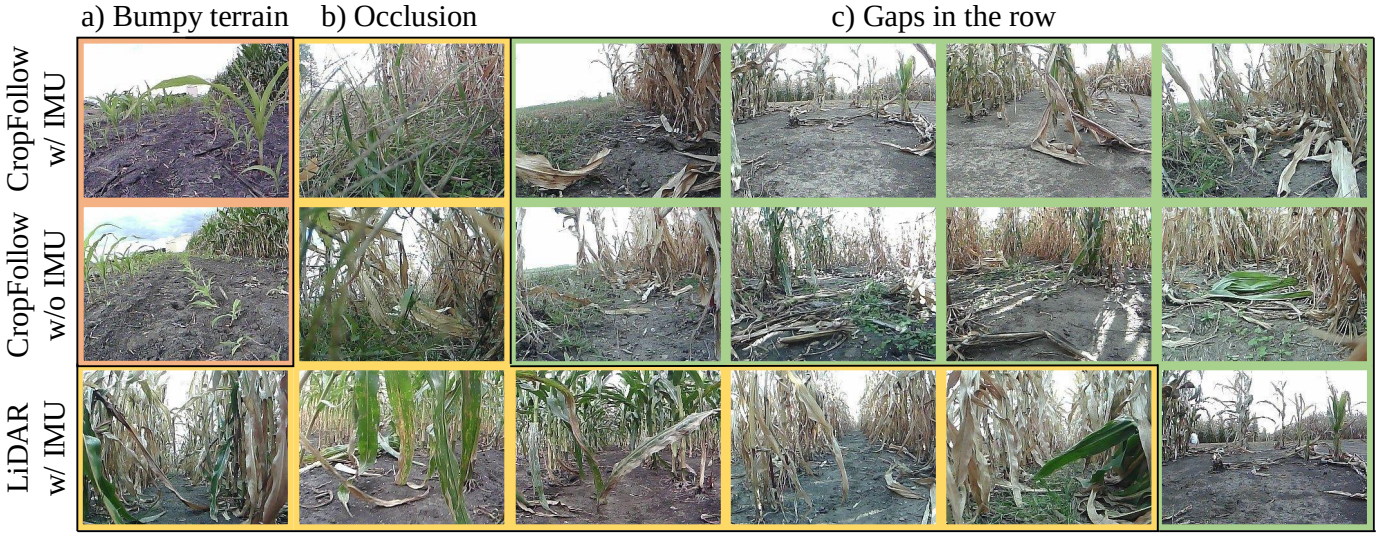

Fig. 14: Failure scenarios for the different navigation systems. We group them into modes: a) bumpy terrain causing noisy, blurry images, b) sensor occlusion from leaves, and c) gaps in crop rows.

navigation systems in challenging application domains, such as agriculture and off-road driving.

VII. Acknowledgments

This paper was supported in part by NSF STTR #1820332, USDA/NSF CPS project #2018-67007-28379, USDA/NSF AIFARMS National AI Institute USDA #020-67021-32799/project accession no. 1024178, NSF IIS #2007035, and DARPA Machine Common Sense. We thank Earthsense Inc. for the robots used in this work and we thank the Department of Agricultural and Biological Engineering and Center for Digital Agriculture (CDA) at UIUC for the Illinois Autonomous Farm (IAF) facility used for data collection and field validation of CropFollow. We thank Vitor Akihiro H. Higuti and Sri Theja Vuppala for their help in integration of CropFollow on the robot and field validation.

References

[1] Diego Aghi, Vittorio Mazzia, and Marcello Chiaberge. Local motion planner for autonomous navigation in vineyards with a rgb-d camera-based algorithm and deep learning synergy. *Machines*, 8(2):27, 2020.

[2] Thomas Bak and Hans Jakobsen. Agricultural robotic platform with four wheel steering for weed detection. *Biosystems Engineering*, 87(2):125–136, 2004.

[3] Tijmen Bakker, Kees van Asselt, Jan Bontsema, Joachim Müller, and Gerrit van Straten. Autonomous navigation using a robot platform in a sugar beet field. *Biosystems Engineering*, 109(4):357–368, 2011.

[4] David Ball, Ben Upcroft, Gordon Wyeth, Peter Corke, Andrew English, Patrick Ross, Tim Patten, Robert Fitch, Salah Sukkarieh, and Andrew Bate. Vision-based obstacle detection and navigation for an agricultural robot. *Journal of field robotics*, 33(8):1107–1130, 2016.

[5] Somil Bansal, Varun Tolani, Saurabh Gupta, Jitendra Malik, and Claire Tomlin. Combining Optimal Control and Learning for Visual Navigation in Novel Environments. In *Conference on Robot Learning*, 2019.

[6] Oscar C Barawid Jr, Akira Mizushima, Kazunobu Ishii, and Noboru Noguchi. Development of an autonomous navigation system using a two-dimensional laser scanner in an orchard application. *Biosystems Engineering*, 96 (2):139–149, 2007.

[7] Marcel Bergerman, Silvio M Maeta, Ji Zhang, Gustavo M Freitas, Bradley Hamner, Sanjiv Singh, and George Kantor. Robot farmers: Autonomous orchard vehicles help tree fruit production. *IEEE Robotics & Automation Magazine*, 22(1):54–63, 2015.

[8] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[9] Chin-Kai Chang, Jiaping Zhao, and Laurent Itti. DeepVP: Deep learning for vanishing point detection on 1 million street view images. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4496–4503. IEEE, 2018.

[10] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Computer Vision and Pattern Recognition*, 2019.

[11] Devendra Singh Chaplot, Saurabh Gupta, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Learning To Explore Using Active Neural Mapping. In *International Conference on Learning Representations*, 2020.

[12] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE international conference on computer vision*, pages

2722–2730, 2015.

[13] Z. Chen and X. Huang. End-to-end learning for lane keeping of self-driving cars. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1856–1860, 2017. doi: 10.1109/IVS.2017.7995975.

[14] Lu Chi and Yadong Mu. Deep steering: Learning end-to-end driving model from spatial and temporal visual cues. *arXiv preprint arXiv:1708.03798*, 2017.

[15] Mang Tik Chiu, Xingqian Xu, Yunchao Wei, Zilong Huang, Alexander G Schwing, Robert Brunner, Hrant Khachatrian, Hovnatan Karapetyan, Ivan Dozier, Greg Rose, et al. Agriculture-vision: A large aerial image database for agricultural pattern analysis. In *Computer Vision and Pattern Recognition*, 2020.

[16] Girish Chowdhary, Eric N. Johnson, Daniel Magree, Allen Wu, and Andy Shein. GPS-denied Indoor and Outdoor Monocular Vision Aided Navigation and Control of Unmanned Aircraft. *Journal of Field Robotics*, 30(3): 415–438, 2013.

[17] Girish Chowdhary, Mattia Gazzola, Girish Krishnan, Chinmay Soman, and Sarah Lovell. Soft Robotics as an Enabling Technology for Agroforestry Practice and Research. *Sustainability*, 11(23):6751, Nov 2019. ISSN 2071-1050. doi: 10.3390/su11236751.

[18] L Cordesses, Christian Cariou, and M Berducat. Combine harvester control using real time kinematic GPS. *Precision Agriculture*, 2(2):147–161, 2000.

[19] Samyak Datta, Oleksandr Maksymets, Judy Hoffman, Stefan Lee, Dhruv Batra, and Devi Parikh. Integrating Egocentric Localization for More Realistic Point-Goal Navigation Agents. volume abs/2009.03231, 2020.

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. IEEE, 2009.

[21] Jay Farrell. *Aided navigation: GPS with high rate sensors*. McGraw-Hill, Inc., 2008.

[22] Jonathan A Foley, Navin Ramankutty, Kate A Brauman, Emily S Cassidy, James S Gerber, Matt Johnston, Nathaniel D Mueller, Christine OaConnell, Deepak K Ray, Paul C West, Christian Balzer, Elena M. Bennett, Stephen R. Carpenter, Jason Hill, Chad Monfreda, Stephen Polasky, Johan Rockstram, John Sheehan, Stefan Siebert, David Tilman, and David P. M. Zaks. Solutions for a cultivated planet. *Nature*, 478(7369):337–342, 2011.

[23] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2009.

[24] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. Learning to fly by crashing. In *IROS*, 2017.

[25] Tianshuang Gao, Hamid Emadi, Homagni Saha, Jiaoping Zhang, Alec Lofquist, Arti Singh, Baskar Ganapathysubramanian, Soumik Sarkar, Asheesh K Singh, and Sourabh Bhattacharya. A novel multirobot system for plant phenotyping. *Robotics*, 7(4):61, 2018.

[26] Iván D García-Santillán, Martín Montalvo, José M Guerrero, and Gonzalo Pajares. Automatic detection of curved and straight crop rows from images in maize fields. *Biosystems Engineering*, 156:61–79, 2017.

[27] Yili Gu, Zhiqiang Li, Zhen Zhang, Jun Li, and Liqing Chen. Path Tracking Control of Field Information-Collecting Robot Based on Improved Convolutional Neural Network Algorithm. *Sensors*, 20(3):797, 2020.

[28] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2017.

[29] Alexandru Gurghian, Tejaswi Koduri, Smita V Bailur, Kyle J Carey, and Vidya N Murali. Deeplanes: End-to-end lane position estimation using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 38–45, 2016.

[30] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[32] Vitor AH Higuti, Andres EB Velasquez, Daniel Varela Magalhaes, Marcelo Becker, and Girish Chowdhary. Under canopy light detection and ranging-based autonomous navigation. *Journal of Field Robotics*, 36(3):547–567, 2019.

[33] Santosh A Hiremath, Gerie WAM Van Der Heijden, Frits K Van Evert, Alfred Stein, and Cajo JF Ter Braak. Laser range finder model for autonomous navigation of a robot in a maize field using a particle filter. *Computers and Electronics in Agriculture*, 100:41–50, 2014.

[34] Guoquan Jiang, Zhiheng Wang, and Hongmin Liu. Automatic detection of crop rows based on multi-ROIs. *Expert systems with applications*, 42(5):2429–2441, 2015.

[35] Gregory Kahn, Pieter Abbeel, and Sergey Levine. Badgr: An autonomous self-supervised learning-based navigation system. *arXiv preprint arXiv:2002.05700*, 2020.

[36] Erkan Kayacan, Sierra N Young, Joshua M Peschel, and Girish Chowdhary. High-precision control of tracked field robots in the presence of unknown traction coefficients. *Journal of Field Robotics*, 35(7):1050–1062, 2018.

[37] Erkan Kayacan, Zhongzhong Zhang, and Girish Chowdhary. Embedded High Precision Control and Corn Stand Counting Algorithms for an Ultra-Compact 3D Printed Field Robot. *Proceedings of Robotics: Science and Systems. Pittsburgh, Pennsylvania*, 2018.

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[39] Ashish Kumar, Saurabh Gupta, David Fouhey, Sergey Levine, and Jitendra Malik. Visual Memory for Robust Path Following. In *NeurIPS*, 2018.

[40] Yecheng Lyu, Lin Bai, and Xinming Huang. Road segmentation using cnn and distributed lstm. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2019.

[41] Wyatt McAllister, Joshua Whitman, Allan Axelrod, Joshua Varghese, Girish Chowdhary, and Adam Davis. Agbots 2.0: Weeding Denser Fields with Fewer Robots. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.062.

[42] Xiangyun Meng, Nathan Ratliff, Yu Xiang, and Dieter Fox. Scaling local control to large-scale topological navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 672–678. IEEE, 2020.

[43] Tim Mueller-Sim, Merritt Jenkins, Justin Abel, and George Kantor. The Robotanist: a ground-based agricultural robot for high-throughput crop phenotyping. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3634–3639. IEEE, 2017.

[44] Matthias Müller, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun. Driving policy transfer via modularity and abstraction. In *Conference on Robot Learning*, 2018.

[45] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[46] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 286–291. IEEE, 2018.

[47] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. Agile autonomous driving using end-to-end deep imitation learning. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.056.

[48] Taihú Pire, Martín Mujica, Javier Civera, and Ernesto Kofman. The Rosario dataset: Multisensor data for localization and mapping in agricultural environments. *The International Journal of Robotics Research*, 38(6): 633–641, 2019. doi: 10.1177/0278364919841437.

[49] William Qi, Ravi Teja Mullapudi, Saurabh Gupta, and Deva Ramanan. Learning to Move with Affordance Maps. In *International Conference on Learning Representations*, 2020.

[50] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[51] Josiah Radcliffe, Julie Cox, and Duke M Bulanon. Machine vision for orchard navigation. *Computers in Industry*, 98:165–171, 2018.

[52] John F Reid, Qin Zhang, Noboru Noguchi, and Monte Dickson. Agricultural automatic guidance research in North America. *Computers and electronics in agriculture*, 25(1-2):155–167, 2000.

[53] Stephane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J. Andrew (Drew) Bagnell, and Martial Hebert. Learning Monocular Reactive UAV Control in Cluttered Natural Environments. In *Proceedings of (ICRA) International Conference on Robotics and Automation*, pages 1765 – 1772. IEEE, May 2013.

[54] Fereshteh Sadeghi and Sergey Levine. CAD^{2}RL: Real single-image flight without a single real image. In *RSS*, 2017.

[55] Redmond Ramin Shamshiri, Cornelia Weltzien, Ibrahim A Hameed, Ian J Yule, Tony E Grift, Siva K Balasundram, Lenka Pitonakova, Desa Ahmad, and Girish Chowdhary. Research and development in agricultural robotics: A perspective of digital farming. *International Journal of Agricultural and Biological Engineering*, 11(4):1–14, 2018.

[56] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[57] Bryce Simmons, Pasham Adwani, Huong Pham, Yazeed Alhuthaifi, and Artur Wolek. Training a remote-control car to autonomously lane-follow using end-to-end neural networks. In *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2019.

[58] Adam Stager, Herbert G. Tanner, and Erin E. Sparks. Design and Construction of Unmanned Ground Vehicles for Sub-Canopy Plant Phenotyping, 2019.

[59] Vijay Subramanian, Thomas F Burks, and AA Arroyo. Development of machine vision and laser radar based autonomous vehicle guidance systems for citrus grove navigation. *Computers and electronics in agriculture*, 53 (2):130–143, 2006.

[60] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[61] Naveen Kumar Uppalapati, Benjamin Walt, Aaron Havens, Armeen Mahdian, Girish Chowdhary, and Girish Krishnan. A Berry Picking Robot With A Hybrid Soft-Rigid Arm: Design and Task Space Control. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.027.

[62] AEB Velasquez, VAH Higuti, HB Guerrero, MV Gasparino, DV Magalhães, RV Aroca, and M Becker. Reactive navigation system based on H∞ control system and LiDAR readings on corn crops. *Precision Agriculture*, 21(2):349–368, 2020.

[63] Andres Eduardo Baquero Velasquez, Vitor Akihiro Hisano Higuti, Mateus Valverde Gasparino, Arun Narenthiran Sivakumar, Marcelo Becker, and Girish Chowdhary. Multi-Sensor Fusion based Robust Row Following for Compact Agricultural Robots. In *arXiv*, 2021.

[64] Stavros G Vougioukas. Agricultural robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 2:365–392, 2019.

[65] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv*, pages arXiv–1911, 2019.

[66] Rui Xu, Changying Li, and Javad Mohammadpour Velni. Development of an Autonomous Ground Robot for Field High Throughput Phenotyping. *IFAC-PapersOnLine*, 51 (17):70–74, 2018.

[67] Jinlin Xue, Lei Zhang, and Tony E Grift. Variable field-of-view machine vision based row guidance of an agricultural robot. *Computers and Electronics in Agriculture*, 84:85–91, 2012.

[68] Sierra N Young, Erkan Kayacan, and Joshua M Peschel. Design and field evaluation of a ground robot for high-throughput phenotyping of energy sorghum. *Precision Agriculture*, 20(4):697–722, 2019.

[69] Zhiqiang Zhai, Zhongxiang Zhu, Yuefeng Du, Zhenghe Song, and Enrong Mao. Multi-crop-row detection algorithm based on binocular vision. *Biosystems engineering*, 150:89–103, 2016.

[70] Chi Zhang and Noboru Noguchi. Development of a multi-robot tractor system for agriculture field work. *Computers and Electronics in Agriculture*, 142:79–90, 2017.

[71] Ji Zhang, George Kantor, Marcel Bergerman, and Sanjiv Singh. Monocular visual navigation of an autonomous vehicle in natural scene corridor-like environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3659–3666. IEEE, 2012.

[72] Qin Zhang, John F Reid, and Noboru Noguchi. Agricultural vehicle navigation using multiple guidance sensors. In *Proceedings of the international conference on field and service robotics*, pages 293–298. August, 1999.

[73] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3357–3364. IEEE, 2017.

[74] Qin Zou, Hanwen Jiang, Qiyu Dai, Yuanhao Yue, Long Chen, and Qian Wang. Robust lane detection from continuous driving scenes using deep neural networks. *IEEE transactions on vehicular technology*, 69(1):41–54, 2019.

Supplementary Material

VIII. Video

Please see the accompanying video that provides an overview of our paper, and shows video executions of our robots. Video is encoded via H-264 MPEG4 and was tested to play well on Windows and MacOS through all regular media players such as Movies & TV (Windows), QuickTime, VLC, and Google Chrome.

IX. IMU Fusion with Extended Kalman Filter

An Extended Kalman Filter was used to reduce the effect of uncertainties in distance and heading estimations. The state vector was defined as $s = (\, d_L \;\; d_R \;\; \phi \;\; \omega \,)^T$ where ϕ is the robot's heading, ω is the angular velocity of the robot and d_R and d_L are the distances to right and left rows respectively. The process was modeled using Eq. 1 where actual state $s[k]$ was defined as a function of $f(\cdot)$ (shown in Eq. 2), the control inputs u_k and the previous state $s[k-1]$,

$$\begin{aligned} s[k] &= f(s[k-1], u[k]) + \omega_k \\ z[k] &= s[k] + \nu_k \end{aligned} \tag{1}$$

here, $f(s[k-1], u[k])$ is defined as,

$$= \begin{bmatrix} d_L[k] \\ d_R[k] \\ \phi[k] \\ \omega[k] \end{bmatrix} = \begin{bmatrix} d_L[k-1] - v\sin(\phi[k-1])\Delta t \\ d_R[k-1] + v\sin(\phi[k-1])\Delta t \\ \phi[k-1] + \omega\Delta t \\ \omega[k] \end{bmatrix}. \tag{2}$$

Both process noise ω_k and measurement noise ν_k were defined as zero mean Gaussian noises and their covariances are $[\,0.001\;0.001\;0.01\;0.01\,]$ and $[\,0.05\;0.05\;0.05\;0.5\,]$ respectively, corresponding to the states and measurements d_L, d_R, ϕ, and ω. Those values are in the covariance matrices $\mathbf{Q}$ (for ω_k) and $\mathbf{R}$ (for ν_k).

The robot's linear v and angular ω velocities are used to estimate the states (Eq. 2) in the prediction step. v is calculated from encoders and ω is obtained from IMU. In the update step, innovation occurs by considering the calculated values of d_L, d_R and ϕ from 2 CNN networks. As the output of the distance CNN network is a distance ratio, it is necessary to convert it to a metric value by multiplying it with average lane width.

X. Model Predictive Control

The kinematic differential model is formulated for a skid-steering mobile robot as presented in Eq. 3.

$$\begin{aligned} \dot{x} &= v\ cos(\phi) \\ \dot{y} &= v\ sin(\phi) \\ \dot{\phi} &= \omega \end{aligned} \tag{3}$$

The robot's states (x, y, ϕ) denote its bi-dimensional position and yaw angle, while inputs v and ω denote its linear and angular speeds. Then, it's possible to transform the differential model to a discrete model and solve it using Euler's integration, as given by Eq. 4.

$$\begin{aligned} x[k] &= x[k-1] + v\ cos(\phi)\ \Delta t \\ y[k] &= y[k-1] + v\ sin(\phi)\ \Delta t \\ \phi[k] &= \phi[k-1] + \omega\ \Delta t \end{aligned} \tag{4}$$

A transformation of $v\ \Delta t = \Delta s$ is adopted, and therefore $\omega\ \Delta t = \rho\ \Delta s$, where ρ is the robot's instantaneous curvature. The new non-linear system is then given by Eq. 5.

$$\begin{aligned} x[k] &= x[k-1] + cos(\phi)\ \Delta s \\ y[k] &= y[k-1] + sin(\phi)\ \Delta s \\ \phi[k] &= \phi[k-1] + \rho\ \Delta s \end{aligned} \tag{5}$$

The non-linear model is used as dynamic model of the process to predict future states, and a cost function over the receding horizon is the optimization cost function using as control input the curvature ρ.

$$\begin{aligned} \min_{\rho_i} \Bigg\{ & \sum_{i=1}^{N} w_{d_{e,i}} d_{e,i}^2 + \sum_{i=1}^{N} w_{\phi_i} \phi_{error,i}^2 + \\ & \sum_{i=1}^{N-2} w_{\Delta\rho_i} (\rho_i - \rho_{i-1})^2 \Bigg\} \end{aligned} \tag{6}$$

Where $d_{e,i}$ is the cross-track error, $\phi_{error,i}$ is the heading error, ρ_i is the curvature command, $w_{d_{e,i}}$ is the weighting coefficient reflecting the relative importance of $d_{e,i}$, w_{ϕ_i} is the weighting coefficient reflecting the relative importance of ρ_i, and $w_{\Delta\rho_i}$ is the weighting coefficient penalizing relative big changes in curvature.

The variables used in the Eq. 6 are calculated as shown in Eq. 7, which shows the calculation of the cross-track error using geometry and the heading error as the difference between current robot's heading (equal to zero in the local robot's frame) and the desired path's heading. The optimization problem is subject to a single constraint inequality acting in the curvature command, which must lay in $-1/R_{max} \leq \rho_i \leq 1/R_{max}$. The R_{max} is the maximum permissible curve radius the robot can follow, which is a tunable parameter for compromise between aggressiveness and to avoid robot to get stuck on difficult terrains.

$$\begin{aligned} \phi_{wp_i} &= \arctan 2(wp_{y_i} - wp_{y_{i-1}}, wp_{x_i} - wp_{x_{i-1}}) \\ R_U &= \sqrt{(wp_{x_{i-1}} - x_i)^2 + (wp_{y_{i-1}} - y_i)^2} \\ \phi_U &= \arctan 2(y_i - wp_{y_{i-1}}, x_i - wp_{x_{i-1}}) \\ d_{e,i} &= R_U\ \sin(\phi_{wp} - \phi_U) \\ \phi_{error,i} &= \arctan 2(wp_{y_i} - wp_{y_{i-1}}, wp_{x_i} - wp_{x_{i-1}}) \end{aligned} \tag{7}$$

In Equation 7, wp_{y_i} and wp_{x_i} are the coordinates of the ith waypoint used as input in the MPC horizon. These waypoints

are generated as a straight line that represents the middle of the crop row, whre this line is calculated from the distance ratio d, lane width W, and the estimated heading ϕ estimated from the vision algorithm and EKF. For each iteration, the cross-track error $d_{e,i}$ and the heading error $\phi_{error,i}$ are calculated as shown in Equation 7, and they are used as functions for the minimization in Equation 6.

During the experiments, the parameters were used as described in Equation 8.

$$\begin{aligned} R_{min} &= 0.7 \\ \Delta s &= 0.2 \\ w_{d_{e,i}} &= \begin{cases} 120 & i = 1, 2, \ldots, 19 \\ 1200 & i = 20 \end{cases} \\ w_{\phi_i} &= \begin{cases} 100 & i = 1, 2, \ldots, 19 \\ 1000 & i = 20 \end{cases} \\ w_{\Delta\rho_i} &= 1000 \quad i = 1, 2, \ldots, 20 \end{aligned} \tag{8}$$

A PID controller is used as low-level controller to guarantee the predicted control effort is followed by the robot. The low-level controller uses the IMU's yaw angular speed as feedback to follow the angular speed command that comes from the MPC controller. As input to the MPC controller, a waypoints generator algorithm is used. The generated path is always straight and built in relation to the robot's local frame. The distance and angle of the straight line depends on the measured distance error and heading error from desired path. Figure 15 shows the overall control diagram, where ω_{MPC} is the yaw angular speed calculated from MPC algorithm, ω_{gyro} is the robot's yaw angular speed measured using the IMU's gyroscope, ω_{error} is the difference between the ω_{MPC} and ω_{gyro}, and ω_{cmd} is the commanded yaw angular speed that is sent to the motors.

XI. Ground Truthing

We use projective geometry to obtain the heading ϕ and distance ratio d from our obtained annotations. We show the derivations for the different steps in this section. We assume a pinhole camera model, and assume camera's focal length to be f. We denote world coordinates with a capital letters (X, Y, Z), and denote their projection in the image as (x, y). Note under the pin hole camera model, $x = \frac{fX}{Z}$ and $y = \frac{fY}{Z}$. The X-axis goes right from the image center, Y-axis goes down from the image center, and the Z axis goes into the scene from the camera center.

As noted our ground truthing process has 4 steps: camera roll correction (Section XI-B), camera pitch correction (Section XI-C), heading estimation (Section XI-D) and distance ratio estimation (Section XI-E). We do these on top of annotated horizon and crop row vanishing lines. For early season data, we can directly annotate these. For late season data, the horizon is not directly visible, and we mark the corn stalks to estimate the horizon from the vanishing point of the vertical lines (Section XI-A). Our annotation procedure assumes: a) ground is flat, b) corn has been planted in parallel rows, c) corn stalks are vertical. We found these to be reasonable assumptions for the data that we were working with.

Images are rotated about the camera center by the obtained roll, pitch and heading angles incrementally between steps, using homography $H = KRK^{-1}$, where K is the camera matrix, and R is the desired rotation about the camera center.

A. Horizon Estimation

Let's assume the vertical stem lines in the world are in the direction (D_X, D_Y, D_Z). Vanishing point of these lines is can be obtained by considering a line in this direction through a point (A_X, A_Y, A_Z), projecting points on this line onto the image plane, and then taking the limit as the points tend to infinity,

$$\begin{aligned} &\lim_{\lambda\to\infty} \left(f_X \frac{A_X + \lambda D_X}{A_Z + \lambda D_Z}, f_Y \frac{A_Y + \lambda D_Y}{A_Z + \lambda D_Z} \right) \\ &= \left(f\frac{D_X}{D_Z}, f\frac{D_Y}{D_Z} \right) = (v_x, v_y) \end{aligned} \tag{9}$$

where (V_X, V_Y) is the vanishing point.
Horizontal plane is given by the following equation,

$$\begin{aligned} D_X X + D_Y Y + D_Z Z &= c \\ D_X \frac{fX}{Z} x + D_Y \frac{fY}{Z} + f D_Z &= \frac{fc}{Z} \\ D_X \cdot x + D_Y \cdot y + f D_Z &= \frac{fc}{Z} \end{aligned} \tag{10}$$

The horizon occurs when we go to ∞ on this plane, *i.e.* $\lim_{Z\to\infty}$,

$$\begin{aligned} D_X \cdot x + D_Y \cdot y + f D_Z &= 0 \\ \frac{D_X}{D_Z} \cdot x + \frac{D_Y}{D_Z} \cdot y + f &= 0 \end{aligned} \tag{11}$$

We can substitute for $\frac{D_X}{D_Z}$ and $\frac{D_Y}{D_Z}$ from Eq. 9, to obtain the equation of the horizon as,

$$v_x \cdot x + v_y \cdot y + f^2 = 0 \tag{12}$$

Vanishing points for the vertical lines can be found by finding the point of intersection of the lines using least squares.

B. Roll Estimation

Suppose the camera is pitched down by pitch θ and has roll α, then the surface normal of the ground plane is given as $(\sin\alpha\cos\theta, \cos\alpha\cos\theta, -\sin\theta)$. The equation of the ground plane is given by:

$$\begin{aligned} &X\cos\theta\sin\alpha + Y\cos\alpha\cos\theta - Z\sin\theta = c \\ \Rightarrow\ &\frac{fX}{Z}\cos\theta\sin\alpha + \frac{fY}{Z}\cos\alpha\cos\theta - f\sin\theta = \frac{fc}{Z} \\ \Rightarrow\ &x\cos\theta\sin\alpha + y\cos\alpha\cos\theta - f\sin\theta = \frac{fc}{Z} \end{aligned} \tag{13}$$

where in the last step, we have substituted image coordinates, x for $\frac{fX}{Z}$ and y for $\frac{fY}{Z}$, using the pinhole camera model. We get the equation of the horizon in the image plane, by seeing what happens as $Z \to \infty$:

$$x\cos\theta\sin\alpha + y\cos\alpha\cos\theta - f\sin\theta = 0 \tag{14}$$

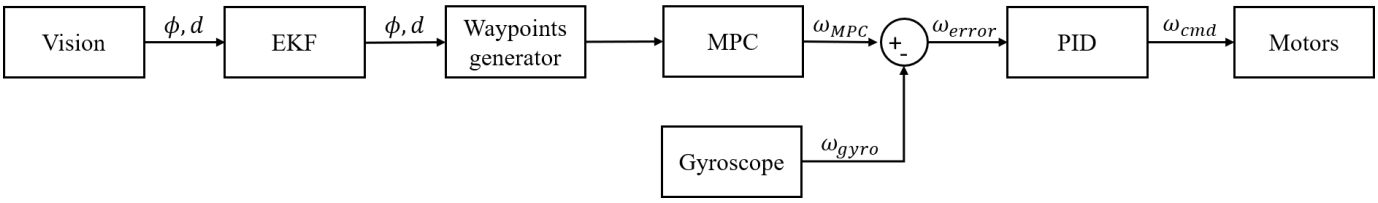

Fig. 15: Overall control diagram.

We can compute the camera roll α from the slope of the horizon h' in the image,

$$\begin{aligned} h' &= -\frac{\cos\alpha\cos\theta}{\sin\alpha\cos\theta} = -1/\tan\alpha \\ \Rightarrow \alpha &= -\arctan(1/h') \end{aligned} \tag{15}$$

C. Pitch Estimation

Assuming the image and all annotation lines have been corrected for roll, we notice that the ground plane will have a normal vector $(0, \cos\theta, -\sin\theta)$. Following a similar procedure as in Section XI-B, we obtain the equation of horizon as:

$$\begin{aligned} 0\cdot X + Y\cos\theta - Z\sin\theta &= c \\ \Rightarrow 0\cdot\frac{fX}{Z} + \frac{fY}{Z}\cos\theta - f\sin\theta &= \frac{cf}{Z} \\ \Rightarrow 0\cdot x + y\cos\theta - f\sin\theta &= \frac{cf}{Z} \end{aligned} \tag{16}$$

As we tend $Z \to \infty$,

$$\begin{aligned} 0\cdot x + y\cos\theta - f\sin\theta &= 0 \\ \Rightarrow y &= f\tan\theta \end{aligned} \tag{17}$$

Thus, θ can be found using the y-intercept of the horizon in the image, as $\arctan(y_{\text{horizon}}/f)$, where y_{horizon} is the y-coordinate of the horizontal horizon line.

D. Heading Estimation

Now that the image and annotation lines have been corrected for pitch and roll, heading ϕ can be obtained from the vanishing point of the crop row lines. The left crop row is in the direction of $(\sin\phi, 0, \cos\phi)$, and let us assume that it passes through the point (A_X^l, A_Y^l, A_Z^l). A point on the the left crop row is given by:

$$(A_X^l, A_Y^l, A_Z^l) + \lambda(\sin\phi, 0, \cos\phi) \tag{18}$$

for different values of λ. These points project to the image place at locations,

$$\left(f\frac{A_X^l + \lambda\sin\phi}{A_Z^l + \lambda\cos\phi}, f\frac{0}{A_Z^l + \lambda\cos\phi}\right) \tag{19}$$

Vanishing point can be obtained by taking $\lim_{\lambda\to\infty}$,

$$\begin{aligned} &\lim_{\lambda\to\infty}\left(f\frac{A_X^l + \lambda\sin\phi}{A_Z^l + \lambda\cos\phi}, f\frac{0}{A_Z^l + \lambda\cos\phi}\right) \\ &= (f\tan\phi, 0) \end{aligned} \tag{20}$$

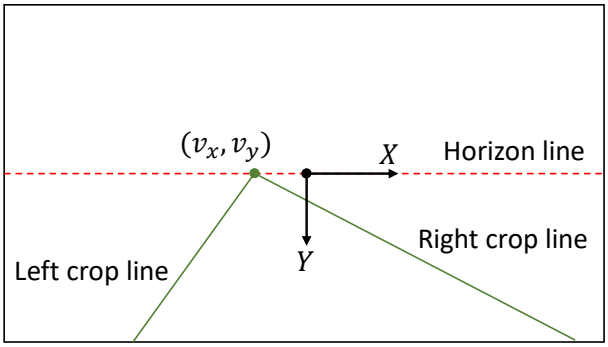

Fig. 16: Shows (v_x, v_y) once the image and annotation lines have been rectified for roll and pitch.

Thus, the heading ϕ can be obtained from the x-coordinate of the vanishing point, v_x as, $\arctan(v_x/f)$. v_x can be obtained from the intersection of the image of the left and the right crop row.

E. Distance Ratio Estimation

Assume that we have rotated the image to correct for the heading. Crop rows are now in the direction $(0, 0, 1)$. Let's assume that the left crop row goes through the point $(X_l, H, 0)$, and the right crop row passes through the point $(X_r, H, 0)$. Then the image of the left crop rows in the image plane is given by:

$$\left(f\frac{X_l + \lambda_l\cdot 0}{0 + \lambda_l\cdot 1}, f\frac{H + \lambda_l\cdot 0}{0 + \lambda_l\cdot 1}\right) = \left(f\frac{X_l}{\lambda_l}, f\frac{H}{\lambda_l}\right) \tag{21}$$

The X-intercept of this line in the image plane is given by the value of λ_l, such that $f\frac{H}{\lambda_l}$ is equal to $h/2$ (where h is the image height), *i.e.*, $\lambda_l = f\frac{H}{h/2}$. Thus, the X-intercept of the left crop row, l_x is $\frac{X_l\cdot h}{2\cdot H}$.

Similarly, the X-intercept for the right crop row, r_x is $\frac{X_r\cdot h}{2\cdot H}$. Thus, the distance ratio, $d = \frac{X_l}{X_l + X_r}$ can be obtained via $\frac{l_x}{l_x + r_x}$.