Towards Automating Precision Irrigation: Deep Learning to Infer Local Soil Moisture Conditions from Synthetic Aerial Agricultural Images

David Tseng*,[1] David Wang*,[1] Carolyn Chen,[1] Lauren Miller,[1] William Song,[1]
Joshua Viers,[2] Stavros Vougioukas,[3] Stefano Carpin,[4] Juan Aparicio Ojea,[5] Ken Goldberg[1,6]

***Abstract*— Recent advances in unmanned aerial vehicles suggest that collecting aerial agricultural images can be cost-efficient, which can subsequently support automated precision irrigation. To study the potential for machine learning to learn local soil moisture conditions directly from such images, we developed a very fast, linear discrete-time simulation of plant growth based on the Richards equation. We use the simulator to generate large datasets of synthetic aerial images of a vineyard with known moisture conditions and then compare seven methods for inferring moisture conditions from images, in which the "uncorrelated plant" methods look at individual plants and the "correlated field" methods look at the entire vineyard: 1) constant prediction baseline, 2) linear Support Vector Machines (SVM), 3) Random Forests Uncorrelated Plant (RFUP), 4) Random Forests Correlated Field (RFCF), 5) two-layer Neural Networks (NN), 6) Deep Convolutional Neural Networks Uncorrelated Plant (CNNUP), and 7) Deep Convolutional Neural Networks Correlated Field (CNNCF). Experiments on held-out test images show that a globally-connected CNN performs best with normalized mean absolute error of 3.4%. Sensitivity experiments suggest that learned global CNNs are robust to injected noise in both the simulator and generated images as well as in the size of the training sets. In simulation, we compare the agricultural standard of flood irrigation to a proportional precision irrigation controller using the output of the global CNN and find that the latter can reduce water consumption by up to 52% and is also robust to errors in irrigation level, location, and timing. The first-order plant simulator and datasets are available at https://github.com/BerkeleyAutomation/RAPID.**

I. Introduction

An estimated 75% of freshwater is used in agriculture, but only between 5% and 30% of this irrigation is absorbed by plants [34]. Environmental scientists predict that water scarcity will be a major issue in the coming decades [35].

In precision agriculture, measurements determine the status of individual plants to modify irrigation and other farm resources accordingly [24]. Recent advances in robotics can facilitate precision irrigation with devices that interact with the plants and irrigation pipelines directly [7, 11] or monitor conditions such as soil moisture [27]. Others take advantage of recent advances in unmanned aerial vehicles (UAVs) [10, 37]. Under the NSF National Robotics Initiative (NRI), the Robot Assisted Precision Irrigation Delivery (RAPID) project investigates how co-robotic devices can be used to adjust irrigation emitters. Prior work in this project has involved developing robotic tools to adjust emitters [4, 5, 11] and routing algorithms for mobile robots [31, 32].

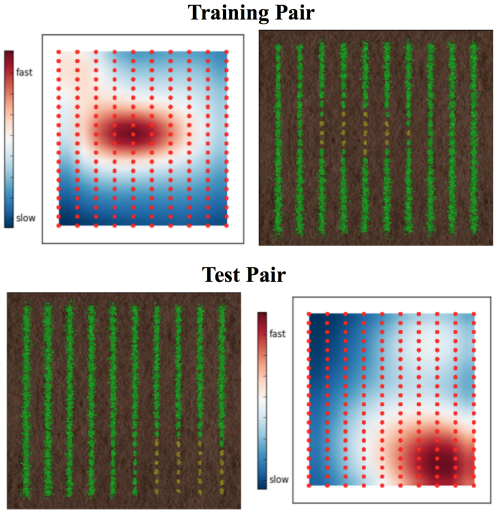

Fig. 1: Top left: Moisture dissipation pattern. Top right: Synthetic aerial image generated by the simulator. We use the simulator to generate 1,200 examples to form a training dataset, and we use support vector machines, random forests, and convolutional neural networks to infer moisture dissipation patterns from images. Bottom left: Example of a test image. Bottom right: Moisture dissipation predicted by the CNN.

In precision irrigation, soil moisture conditions determine the amount of irrigation needed. To the best of our knowledge, there are no publicly available datasets with pairs of aerial images and local soil moisture conditions at the plant level. In this paper, we explore machine learning methods to learn mappings between images and soil moisture dissipation patterns and how the learned models could control precision irrigation at the plant level. An overview is shown in Fig. 1. We study this problem in simulation as a first step to explore the potential of machine learning for precision irrigation before undertaking extremely costly physical field experiments. We present a first-order simulator that models the local soil moisture at each plant over time using a discrete-time, linear approximation of the Richards equation [33] for soil water flow. One major benefit of this first-order simulator is its speed, which allows us to quickly generate datasets of vineyard images based on different soil moisture dissipation patterns. We perform experiments with seven methods to learn

The AUTOLab at UC Berkeley (automation.berkeley.edu).
[1]EECS, UC Berkeley, {davidtseng, dmwang, carolyn.chen, william.song, goldberg}@berkeley.edu, lauren.mohr.miller@gmail.com; [2]Center for Watershed Sciences, UC Merced, jviers@ucmerced.edu; [3]Biological and Agricultural Engineering, UC Davis, svougioukas@ucdavis.edu; [4]School of Engineering, UC Merced, scarpin@ucmerced.edu; [5]Siemens Corporation, Corporate Technology, juan.aparicio@siemens.com; [6]IEOR, UC Berkeley.
* These authors contributed equally to the paper.

the inverse mapping from images to soil moisture dissipation patterns, in which the "uncorrelated plant" methods look at individual plants and the "correlated field" methods look at the entire vineyard: 1) constant prediction baseline, 2) linear Support Vector Machines (SVM), 3) Random Forests Uncorrelated Plant (RFUP), 4) Random Forests Correlated Field (RFCF), 5) two-layer Neural Networks (NN), 6) Deep Convolutional Neural Networks Uncorrelated Plant (CNNUP), and 7) Deep Convolutional Neural Networks Correlated Field (CNNCF). We analyze the robustness of these models by conducting sensitivity experiments, exposing the learning methods to limited numbers of training examples and injecting two types of noise: simulation noise, where we perturb the simulator dynamics, and image noise, where we perturb the generated aerial images. We also implement a proportional precision irrigation controller in simulation using the learned models.

This paper makes the following contributions:

1) A novel first-order agricultural simulator that uses a discrete-time, linear approximation of the Richards equation for soil moisture dissipation.
2) A labeled dataset for supervised learning, in which the examples are pairs of soil moisture dissipation patterns and the generated synthetic aerial images, containing 1200 training examples and 200 test examples.
3) Experiments comparing seven machine learning methods that are able to learn inverse models from aerial images to the underlying soil moisture dissipation patterns and experiments with a proportional precision irrigation controller using the learned globally-connected CNN.

II. Related Work

A. Agricultural Simulators

Several crop simulators have been published in the agricultural community. WOFOST [8], DSSAT [19], and MONICA [26] simulate aggregate crop statistics such as yield based on parameters such as weather and location. Other simulators provide similar information for specific crops, like SUNFLO [6] for sunflowers and MaizSim [38] for corn. Such simulators provide accurate estimates by modeling plant biology and the environment but generally output aggregate crop statistics over entire vineyards rather than on a per-plant basis, which cannot be used to generate labeled data for a precision irrigation model. While our first-order simulator does not explicitly model physical processes at the level of plant biology, such as in [22], it provides reasonable simulated aerial images of a field of crops, given local soil moisture dissipation rates at each of the plants. A comparison between the simulator output and a real vineyard is shown in Fig. 2.

B. Agricultural Image Datasets

To the best of our knowledge, large-scale, public datasets of paired aerial field images and corresponding local soil conditions for individual plants do not exist. Of the existing agricultural image datasets, many are tailored to classification and segmentation experiments. Söderkvist published a dataset of leaves from Swedish trees [29], Wu et al. published the

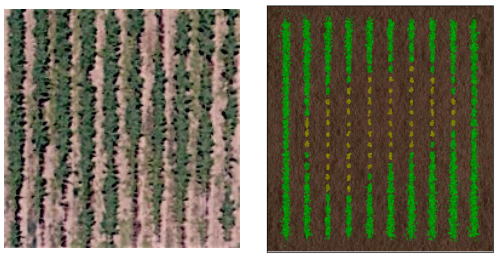

Fig. 2: Example of (left) a true aerial image of a vineyard of Symphony grapes in Cowell Ranch of Snelling, CA and (right) a synthetic aerial image generated by the first-order simulator. There is inherent heterogeneity of plant growth in both.

Flavia dataset of leaves from 33 plant species [36], and Kumar et al. published a combined dataset of high-resolution leaves from the Smithsonian collection and images taken from mobile phones in the field [21]. These labeled datasets are used for plant species identification. Recently, Haug et al. [16] published a labeled dataset of aerial field images for the purpose of segmentation and crop/weed classification. Hassan-Esfahani et al. took aerial images of fields using a UAV [13–15] as part of the AggieAir™ project led by Cal Coopmans and Alfonso Torres Rua. They also made local soil moisture measurements to create a dataset with 184 points, demonstrating the potential to collect a larger dataset. In this paper, we generate a dataset that consists of 1,400 synthetic aerial images of a vineyard and corresponding local soil moisture conditions at each plant.

C. Machine Learning in Agriculture

There exist previous applications of machine learning in agricultural settings to learn soil moisture conditions. Schmitz et al. [28] used neural networks to learn the relationship between irrigation and soil moisture profiles for drip irrigation to optimize soil moisture. Möller et al. [25] utilized visual and thermal images of crops from a Forward-Looking Infrared Radar (FLIR) thermal imaging system to estimate crop water stress index (CWSI) [18]. Hassan-Esfahani et al. estimated topsoil moisture profiles using Relevance Vector Machines, Multilayer Perceptrons, and Bayesian Neural Networks and multiple types of images collected using a UAV [13–15] in the AggieAir™ project. Other applications involve SVMs or RVMs to estimate soil parameters using a variety of data such as weather data and crop physiological characteristics instead of aerial images [2, 3, 39]. In this paper, we present an end-to-end simulation system that generates synthetic aerial images and learns mappings from the images to local soil moisture conditions using both previously used techniques as well as deep CNN's. We also evaluate the effectiveness of precision irrigation controllers using the learned models, which can be more efficient than traditional methods such as flood irrigation. For example, Mateos et al. [23] compared drip irrigation and furrow irrigation, a type of flood irrigation, in cotton crops and used up to 46% less water with the former strategy.

III. Problem Statement and Assumptions

Our goal is to learn a model that takes a synthetic RGB aerial image as input and outputs predictions of soil moisture

dissipation patterns. We first address the forward problem: given soil moisture dissipation patterns, rapidly generate a synthetic 320×320 RGB image of a vineyard containing p plants at t days after the plants are planted, given $\mathbf{d} \in \mathbb{R}^p$, the local soil moisture dissipation rate at each of the plants.

Here, we present modeling assumptions for the design of the simulator and for the prediction of soil moisture dissipation patterns. For the growth of each plant in the vineyard, we use the model of one-dimensional vertical water flow to simulate the dynamics of water flow in soil. This can be expressed with the following nonlinear partial differential equation, which is the basis of the well-known Richards equation [33]:

$$\frac{\partial s}{\partial t} = -\frac{\partial q}{\partial z} - S(h),$$

where s is the volumetric water content in the soil (cm^3 of water per cm^3 of soil), t is the time (days), $\frac{\partial q}{\partial z}$ is a term that describes the flow of soil water as a result of differences in soil water potential, and $S(h)$ is a term that describes the soil water extraction by plant roots (cm^3 of water per cm^3 of soil per day). In this paper, we refer to s as a measure of soil moisture. In the simulator, we approximate this differential equation using the following discrete linear update equation, which is why we refer to the simulator as being first-order:

$$s_t = s_{t-1} + \Delta s_t, \text{ where } \Delta s_t = -(r - a_t) - U(t).$$

Here, r is the soil drainage rate, a_t is the irrigation rate applied during day t, and $U(t)$ is the rate at which plant water uptake occurs during day t. In relation to the Richards equation, we approximate $-\frac{\partial q}{\partial z}$ with $a_t - r$, the rate at which water becomes available for use during day t, and we also have $S(h) = U(t)$. Under optimal moisture conditions, the following equation for $U(t)$ holds [9]:

$$U(t) = R(t) \cdot T_p(t),$$

where $R(t)$ is a factor related to root length density at time t, and $T_p(t)$ is potential transpiration rate (cm per day) that is driven by weather. In the first-order simulator, we assume the following remain constant over time for a particular plant: $R(t)$ (water uptake capacity due to root size does not change as the plant grows), $T_p(t)$ (weather does not change significantly to cause variation in daily plant water transpiration), and r (the local soil drainage rate does not vary). Thus, in the simulator, $U(t)$ is also constant over time, and we simply write U. Under these assumptions, our discrete-time linear update equation can be written as follows:

$$s_t = s_{t-1} + a_t - (r + U) = s_{t-1} + a_t - d.$$

We define $d = r + U$ to be the moisture dissipation rate, which is a measure of local water loss, e.g., due to soil drainage, uptake by the plant, evaporation, etc. We attempt to learn d from the images. We also assume linear plant growth, where each plant grows a constant f new leaves on day t if $s_t > 0$. The simulator is described in further detail in Section IV-A.

The first objective is to use the simulator to generate a dataset containing pairs of ground-truth local soil moisture dissipation rates of the plants $\mathbf{d}$ and the resulting image $\mathbf{x}$. We define $\mathbf{d}^{(i)}$ and $\mathbf{x}^{(i)}$ to be the soil moisture dissipation rates and corresponding image, respectively, for the i-th example in the training dataset for $i = 1$ to N_{train}. Using the same process, we generate a test dataset containing N_{test} examples.

The next objective is to learn the inverse model H_{θ^*} from the images $\{\mathbf{x^{(i)}}\}_{i=1}^{N_{train}}$ to the local soil moisture dissipation rates $\{\mathbf{d^{(i)}}\}_{i=1}^{N_{train}}$ of the plants in the images, where H represents the model, θ represents the parameters of H, and $H_\theta(\mathbf{x})$ represents the predicted soil moisture dissipation rates given by model H_θ on input image $\mathbf{x}$. Ultimately, we seek to find $\theta^* \in \Theta$, where Θ represents the space of parameters θ, such that the expected loss $\mathcal{L}$ over all possible simulated images that can result from the process described in Section IV-B is minimized, i.e.,

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} \, \mathbf{E}[\mathcal{L}(\mathbf{d}, H_\theta(\mathbf{x}))].$$

We want to minimize the mean squared error loss function:

$$\mathcal{L}(\mathbf{d}, H_\theta(\mathbf{x})) = \frac{1}{p} \sum_{j=1}^{p} (\mathbf{d}_j - H_\theta(\mathbf{x})_j))^2.$$

Since it is difficult to minimize the expected loss, we seek θ^* that minimizes the loss over our generated training dataset,

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{i=1}^{N_{train}} \mathcal{L}(\mathbf{d}^{(i)}, H_\theta(\mathbf{x}^{(i)})).$$

IV. First-Order Plant Simulator

A. Discrete Time Simulator

We define a vineyard of length l and width w. We assume n columns of plants uniformly spaced across the vineyard and m individual plants uniformly spaced within each column for a total of mn individual plants.

The simulator generates aerial images of the vineyard at discrete timesteps (days), based on the following parameters and inputs.

- $\mathbf{d} \in \mathbb{R}^{mn}$ is a vector of local soil moisture dissipation rates for each of the plants, measuring the rate of water loss due to a combination of soil drainage and uptake by the plant itself. These rates do not vary over time.
- $\mathbf{a^{(t)}} \in \mathbb{R}^{mn}$ is a vector of applied irrigation rates for each plant on day t for $t = 1, 2, \ldots$.

In this model, the primary factor for determining plant growth is soil moisture. Letting $\mathbf{s^{(t)}} \in \mathbb{R}^{mn}$ denote a vector of local soil moistures for each of the plants during day t, these vectors are initialized and updated as follows, always ensuring that the soil moisture is non-negative:

$$\mathbf{s^{(0)}} = \mathbf{1} \tag{IV.1}$$

$$\mathbf{s^{(t)}} = \max(\mathbf{s^{(t-1)}} + \mathbf{a^{(t)}} - \mathbf{d}, \mathbf{0}) \text{ for } t = 1, 2, \ldots \tag{IV.2}$$

In each timestep, the visible growth of the plant depends on the current soil moisture value. If the soil moisture is positive, a fixed number of leaves f is added to each plant. The location of the new leaves are sampled from a multivariate

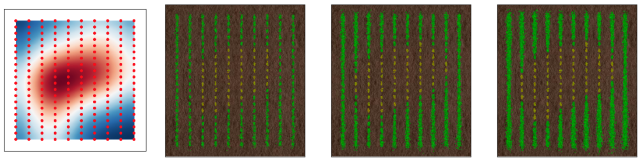

Fig. 3: (left) We show the soil moisture dissipation rate map of the vineyard generated using the process described in Section IV-B. Red indicates higher dissipation rates, and blue indicates lower rates. A constant irrigation rate is applied at each time step, and we show the aerial image generated by the simulator in subsequent images at timesteps 5, 10, and 15, respectively. Note that only the images from the 10th timestep are used in the training and test datasets.

normal distribution with a mean at the center of the plant and a covariance matrix proportional to the current soil moisture. If the soil moisture is 0, then the plant is considered underwatered, no leaves are added, and the existing leaves are colored by a shade of yellow.

For experiments, we use the following simulator parameter values: $l = 100$ and $w = 100$ for the vineyard dimensions, $m = 20$ and $n = 10$ for the number of plants, and $f = 7$ for the number of leaves added per plant in each timestep. Fig. 3 shows an example of the simulation output.

B. Dataset Generation

To generate the dataset, we apply a constant irrigation rate of 0.5 to every plant at each timestep, and we generate an independent soil moisture dissipation map for each example. To simulate variability in soil moisture dissipation rates due to variations in soil, elevation, sun, and wind, we generate dissipation rates for each example in the dataset according to the following process:

1) Choose the number of areas k of high dissipation rate uniformly at random between 2, 3, and 4.
2) We model each of the areas of high dissipation rate as a multivariate normal distribution. In the vineyard of length l and width w, we choose the center $\mu^{(i)} \in \mathbb{R}^2$ for each of the k areas of high dissipation rate uniformly at random.
3) We randomize a diagonal covariance matrix, $\Sigma^{(i)}$, for each distribution. We assume independence between the two directions and select standard deviations in both directions uniformly at random in ranges defined by $\min_{\sigma_x}$, $\max_{\sigma_x}$, $\min_{\sigma_y}$, and $\max_{\sigma_y}$,
4) After defining these areas of high dissipation rate, we take a linear combination of the distributions to obtain a density $d_e(x,y)$ of effective dissipation rate over the entire vineyard, i.e.

$$d_e(x,y) = \sum_{i=1}^{k} c \cdot \exp\left(-\frac{(x-\mu_1^{(i)})^2}{2{\Sigma_{1,1}^{(i)}}^2} - \frac{(y-\mu_2^{(i)})^2}{2{\Sigma_{2,2}^{(i)}}^2} \right).$$

We use $c = 0.4$ as a scaling factor so that each multivariate normal distribution attains a maximum value of c. We normalize so that the local soil moisture dissipation rate of a plant at location (x,y) in the vineyard is $d(x,y) = \min(d_e(x,y), 1)$.

For each example in the dataset, we use the process outlined above to generate the soil moisture dissipation map, and we run the simulator to obtain the corresponding aerial image. For the ranges of standard deviations in the multivariate normal distributions, we use $\min_{\sigma_x} = 10$, $\max_{\sigma_x} = 40$, $\min_{\sigma_y} = 10$, and $\max_{\sigma_y} = 40$. Based on visual inspection, we decided to use the generated aerial image at timestep 10 because, at this point, the image appears most representative of true images of vineyards. We generated 1,200 independent examples for the training set and 200 independent examples for the test set, which took 3 seconds per example on a machine with an Intel i7 Core Processor and 16 GB of RAM. The leftmost image in Fig. 3 shows an example generated from this process.

V. Learning the Inverse Model

We consider seven methods of learning the inverse model H_θ: 1) constant prediction baseline, 2) linear Support Vector Machines (SVM), 3) Random Forests Uncorrelated Plant (RFUP), 4) Random Forests Correlated Field (RFCF), 5) two-layer Neural Networks (NN), 6) Deep Convolutional Neural Networks Uncorrelated Plant (CNNUP), and 7) Deep Convolutional Neural Networks Correlated Field (CNNCF). Each method produces a candidate inverse model that takes in a simulated aerial image as input and outputs the corresponding estimated moisture dissipation rates, and each candidate inverse model has its own separate set of parameters. In addition, for each method, we use the same training set to tune the model's parameters and the same testing set to evaluate the metrics described in Section VI-A. We also randomly split the initial training dataset into a training set with 1,000 examples and a validation set with 200 examples in our methods to tune hyperparameters.

We evaluated two variants of some of our methods, comparing learning dissipation rates for the entire vineyard simultaneously vs. learning rates for individual plants. For the "uncorrelated" methods, which look at the vineyard on a plant-by-plant basis and predict moisture dissipation rates for each plant independently from neighboring plants, the simulated images were preprocessed by cropping individual plants from the original vineyard images. On the other hand, the "correlated" methods look at the entire image of the vineyard and output a vector of 200 dissipation rates.

A. Constant Prediction Baseline

As a baseline for comparison, this method produces a model that outputs a constant prediction c, the mean of all of the moisture dissipation rates in the training set.

B. Linear Support Vector Machine Regression (SVM)

We trained 200 linear support vector machine (SVM) regressors using the RGB values of the images as features, where each SVM learns the moisture dissipation rate at a particular plant location. This was implemented using scikit-learn. Since the number of features is significantly greater than the number of training examples, the dimensionality is already high, and using more complex kernels would impose additional computation time. Thus, we use a linear kernel instead of a radial basis function (RBF) kernel [17].

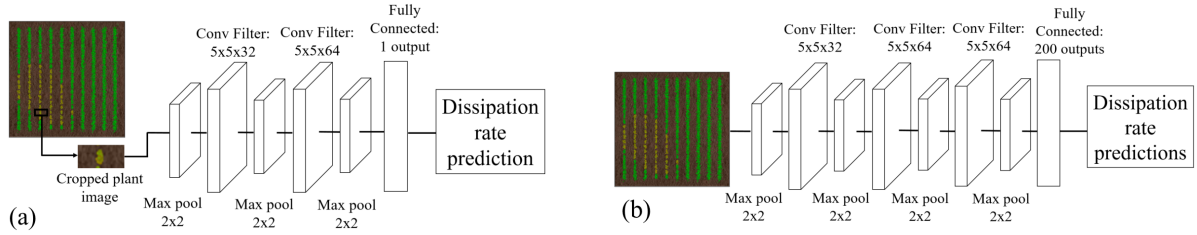

Fig. 4: The CNN architectures for the methods described in Sec. V-F and V-G. (a) The CNN Uncorrelated Plant Method. Each individual plant image is cropped from the original simulated vineyard image and is input to the above CNN. (b) The CNN Correlated Field Method. The simulated vineyard image is input to the above CNN, and the moisture dissipation rates for all 200 plants are estimated together.

C. Random Forest Uncorrelated Plant Method (RFUP)

In this method, we look at images of individual plants with a random forest, using the image RGB values as features. The simulated images are preprocessed using the cropping method mentioned in the beginning of Section V before being given to a RF as input. The RF was implemented with 10 decision trees using scikit-learn with a maximum depth of 20, and parameters were determined using the validation set.

D. Random Forest Correlated Field Method (RFCF)

This method is similar to that of Section V-C, except the RF trains on images of the entire vineyard. The RF implementation is the same as the one described in Section V-C.

E. Two-Layer Neural Network (NN)

We implemented a two-layer feed-forward NN to serve as comparison with the convolutional NN architectures described in Sections V-F and V-G. It is also similar to the architecture used by [14] in the AggieAir™ project, but not directly comparable. In this method, aerial images are preprocessed using the cropping process described earlier. The RGB values of the cropped plant images are provided as input into the NN, which consists of a hidden layer with 4096 units, ReLU activations [12], and an output layer with a single node containing the moisture dissipation prediction. The estimated dissipation rates for individual plant images are concatenated together to form an output with 200 predictions. In this TensorFlow [1] implementation, we perform training with dropout [30] of probability 0.25 for regularization, Kingma et al.'s Adam [20] optimization with an initial learning rate of 1×10^{-3}, and 40 epochs of learning. We use the validation set to tune hyperparameters.

F. CNN Uncorrelated Plant Method (CNNUP)

In this method, the images are cropped and provided as input to a CNN, similar to Section V-E. The CNN architecture used in this method is illustrated in Fig. 4a. The CNN was implemented using TensorFlow [1], the initial learning rate for the Adam algorithm was set to 1×10^{-3}, and we used mini-batches of 25 images, which were values that we observed lead to relatively quick convergence. We optimized the parameters θ in the CNN by applying Adam algorithm and used ReLU activation, dropout of probability 0.25, and training for 40 epochs. We use our CNN's performance on the validation set to tune hyperparameters.

G. CNN Correlated Field Method (CNNCF)

This method is similar to the CNN Uncorrelated Plant Method in Sec. V-F, except that the whole image is used as input. The architecture used for this method is described in Fig 4b. Compared to the architecture used in V-F, CNNCF uses an additional convolutional layer and max pool layer, since the input is larger. The implementation and optimization are the same, but we use an initial learning rate 1×10^{-4} and train for 30 epochs.

Method	Q_1	Mean	M	Q_3	Training Time (s)
Const. Prediction	0.082	0.177	0.162	0.246	N/A
SVM	0.015	0.038	0.030	0.053	4141
RFUP	0.030	0.071	0.060	0.101	578
RFCF	0.047	0.109	0.094	0.152	655
Two-Layer NN	0.037	0.086	0.075	0.119	**243**
CNNUP	0.021	0.054	0.044	0.077	538
CNNCF	**0.012**	**0.034**	**0.027**	**0.047**	603

TABLE I: Results from applying each method to the test dataset. We report the following statistics over all absolute value prediction errors in the test dataset: 25th percentile (Q_1), mean, median (M), and 75th percentile (Q_3). We also report the training time (s) for each of the methods on a machine with an Intel Core i7-6850k Processor and three Nvidia Titan X Pascal GPU's.

VI. Experiments

A. Evaluation Metrics

We use the median absolute error as our main evaluation metric because of its interpretability as the normalized prediction error. We define the absolute error for a single moisture dissipation rate as $|\mathbf{d}_j - H(\mathbf{x})_j|$, where $H(\mathbf{x})_j$ is the estimated moisture dissipation rate of plant j in input image $\mathbf{x}$, and $\mathbf{d}_j$ is the actual dissipation rate of plant j. We report the median, since our distributions of absolute value errors tend to be skewed; in the simulator, plants with higher soil moisture dissipation rates have less variability in appearance than plants with lower dissipation rates. As a result, all models tend to have higher absolute error when predicting higher dissipation rates. We report the interquartile range of the absolute errors of all moisture dissipation rates in the test set to show the distribution.

B. Estimating Moisture Dissipation Rates in the Test Dataset

We evaluated the performance of each method on the test dataset. Each method was trained on the same training and validation set, with no added noise. Table I lists the resulting median and mean, as well as the first and third quartiles, of the absolute errors in the test dataset. We found that CNNCF

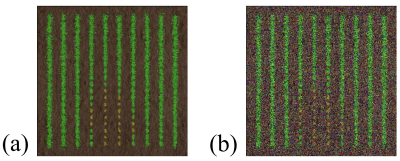

Fig. 5: Example of the effect of adding Gaussian noise directly to the simulated image. (a) The original image. (b) The image after adding $\sim \mathcal{N}(0, 63.75^2)$ noise to all three channels.

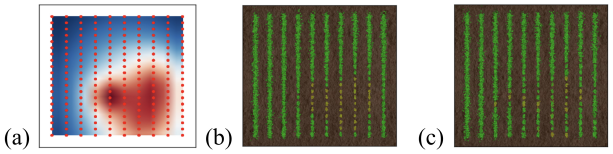

Fig. 6: Example of the effects of adding Gaussian noise to Eq. IV.2. (a) Depicts the ground-truth moisture dissipation rate map, (b) depicts the corresponding vineyard after 10 timesteps with no simulation noise, and (c) depicts the corresponding vineyard after 10 timesteps with added $\sim \mathcal{N}(0, 0.25^2)$ simulation noise.

had the lowest median absolute error (0.034) and smallest interquartile range. This demonstrates the effectiveness of CNN architectures, which take into account the structure of the input as an image. Furthermore, CNNCF performs better than CNNUP because it has access to all plants, which have correlated moisture conditions. However, both CNNCF and CNNUP perform better than the baseline neural network approach (NN). The Linear SVM also performed well, achieving a median error of 0.038. We believe this is due to the linear growth model used in the simulator, which can be effectively captured by a linear function. Finally, RFUP performs better than RFCF. The learning problem requires an output in high-dimensional space, so more data is needed for RFCF to improve performance.

C. Robustness of Models to Injected Image Noise

We tested the robustness of the trained models to images injected with RGB noise. We generated three additional datasets by adding zero-mean Gaussian noise with standard deviations listed in Fig. 7a. to all three channels of images from the original training set. These standard deviation values correspond to fractions (0.25, 0.5, and 0.75) of the range of RGB values. An example of an image with added RGB noise is shown in Fig. 5b. The results are shown in Fig. 7a. We observed that the absolute errors for all trained models increased with noise level, but at each noise level, CNNCF had the lowest median absolute error: 0.027, 0.097, 0.010, and 0.112. Even though the SVM produced a good model on the noiseless dataset, it was most sensitive to injected image noise, having the greatest error at the two highest levels of noise.

D. Robustness of Models to Injected Soil Measurement Noise

To simulate noise and uncertainty in soil measurements, we added *i.i.d.* zero-mean Gaussian noise to the soil moisture update in Eq. IV.2 at each timestep. We generated four additional datasets with noise, each corresponding to a different standard deviation of the added noise: 0.25, 0.5, 0.75, and 1. An example of a resulting vineyard is shown in Fig. 6c. The results are shown in Fig. 7b. The median

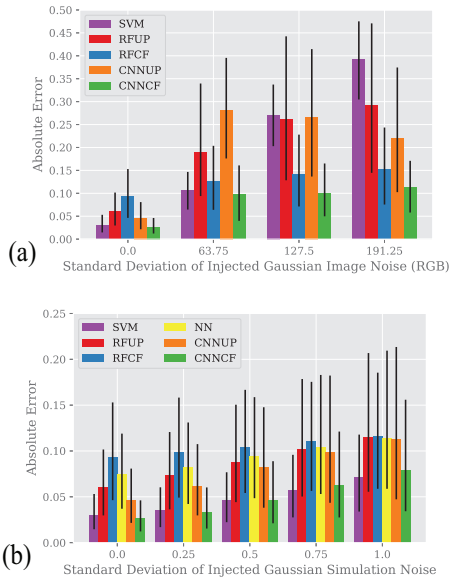

Fig. 7: Injected noise experiments. Plot of the median absolute error (M) of the models when applied to test datasets with (a) injected image noise and (b) injected simulation noise. The error bars represent the 25th and 75th percentiles. Note that the results from the two-layer neural network are omitted in (a), since this method exhibited extremely high sensitivity to injected image noise.

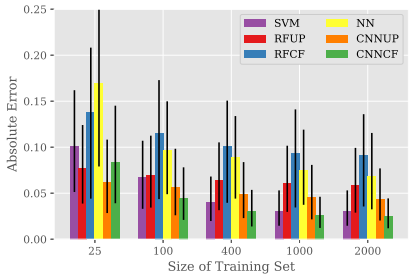

Fig. 8: Quality of models trained on different dataset sizes. Plot of the median absolute error (M) in the test dataset of the resulting models. The error bars represent the 25th and 75th percentiles.

absolute value error of all models increased as the noise level increased, but either CNNCF or SVM had the lowest median absolute error at all noise levels: 0.027, 0.034, 0.047, 0.063, and 0.079. We believe the SVM method is much more sensitive to image noise due to the lack of expressiveness of linear models. While image noise strongly perturbs the input images, simulation noise changes the variation in plant growth but produces reasonable images.

E. Effect of Training Set Sizes on Quality of Models

We also ran experiments to evaluate prediction error when using different training set sizes, shown in Fig. 8. All of the methods produced better models as the training set size increased. However, the methods that utilized entire images (SVM, RFCF and CNNCF) experienced greater improvements compared to the other methods. At a training set size of 100, the median absolute error of CNNCF is already lower than the other methods.

F. Proportional Precision Irrigation Control

In the simulator, plant growth is a function of soil moisture, so to achieve uniform growth of plants over time, we need to vary irrigation so that the moisture profile of the entire field is uniform. An example of a proportional control method is as follows: let $\mathbf{x_t}$ be the aerial image generated at timestep t, and H_θ be our learned model using CNNCF. We use the following update function to proportionally increase and decrease the irrigation rate when the model predicts the moisture dissipation rate to be above and below k, respectively:

$$\mathbf{a^{(t)}} = \mathbf{a^{(t-1)}} + \alpha(H_\theta(\mathbf{x_{t-1}}) - k \cdot \mathbf{1}).$$

We experimentally determined that $\alpha = 1$ and $k = 0.25$ achieved uniform growth of plants over time. An example of the results of applying this controller in the simulator is shown in Fig. 9. This suggests that even though the model was trained using simulation outputs at the tenth timestep with constant irrigation applied at each timestep, it can provide an irrigation policy that effectively corrects for soil variation.

In the simulator, by using the above control strategy to determine irrigation levels for each individual plant in the test dataset, 52% less water is used than when using an example flood irrigation strategy, in which all plants in the field are given the same amount of water, a_{flood}. We found that $a_{flood} = d_{max} + k$, where d_{max} is the largest predicted dissipation rate in the field and $k = 0.25$ as in the feedback loop, allows all plants to grow consistently.

One implementation of a precision irrigation system uses humans or robots to adjust individual irrigation emitters at each plant with a co-robotic device [11]. Above, we considered having perfect and instantaneous emitter adjustments, but any implementation of this system has errors, which we study with the simulator:

- Adjustment Errors: The robot may not adjust the emitter by the desired amount, which is simulated as adding zero-mean Gaussian noise to every adjustment.
- Spatial Errors: The robot may adjust a neighboring emitter instead. We simulate spatial errors by adjusting an adjacent emitter with probability p.
- Time Delays: Adjustments may not occur at every timestep, simulated as adjusting once every t timesteps.

We conduct each of these experiments over the first 20 vineyard examples in the test dataset, allowing the simulation to proceed for the first 10 timesteps before making noisy adjustments over the next 10 timesteps. Since we would like uniform soil moisture throughout the field, we measure the resulting variance of the soil moisture across all 200 plants in the field for each example and report confidence intervals on this metric. The results are shown in Table II. We observe that adding these types of errors increases the resulting soil moisture variance, but they still perform significantly better than the baseline of flood irrigation.

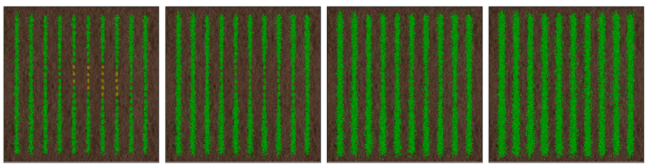

Fig. 9: Precision irrigation control: Example development of the vineyard using a feedback controller based on the CNNCF inverse model's moisture dissipation estimates. The first image is the simulated vineyard after ten timesteps. The field is then irrigated using a feedback controller, and subsequent images are taken after every five timesteps.

Experiment	Mean Soil Moisture Variance
Flood (Baseline)	12.69 ± 2.14
Precision (Baseline)	02.05 ± 0.35
Precision (Adjustment Errors)	04.55 ± 0.49
Precision (Spatial Errors)	02.32 ± 1.02
Precision (Time Delays)	05.08 ± 0.67

TABLE II: We simulate various types of noise in the implementation of a precision irrigation controller and report 95% confidence intervals of the resulting soil moisture variance. The baseline experiments have no added errors. For adjustment errors, we use zero-mean Gaussian noise with standard deviation 0.1. For spatial errors, we use probability $p = 0.3$. For time delays, we incorporate a delay of $t = 3$ timesteps.

VII. Discussion and Future Work

We studied seven methods: 1) constant prediction baseline, 2) linear Support Vector Machines (SVM), 3) Random Forests Uncorrelated Plant (RFUP), 4) Random Forests Correlated Field (RFCF), 5) two-layer Neural Networks (NN), 6) Deep Convolutional Neural Networks Uncorrelated Plant (CNNUP), and 7) Deep Convolutional Neural Networks Correlated Field (CNNCF). In nearly all experiments, the CNN Correlated Field Method had the lowest median absolute test error compared to the other methods. Furthermore, by using the predictions from the most accurate learned model in a proportional precision irrigation controller that adjusts irrigation rates for individual plants, we observe that 52% less water is used compared to flood irrigation and this control is robust to errors in irrigation level, location, and timing.

We acknowledge the many limitations of this first-order plant simulator. It was designed for speed and and does not include many other factors such as temperature, humidity, and wind, as well as variations in lighting, plant genetics, and pests. However, we believe these first-order results and sensitivity analyses suggest that Deep Learning has great potential for applications in precision irrigation.

Deep Learning techniques rely on large and relatively clean datasets. In this case, they require significant monitoring of a vineyard's growing season using well-calibrated and consistent aerial UAV images with (literal) ground truth where soil moisture is measured at each individual plant and well-correlated with the aerial images. This requires careful sensing at many ground points and consistent UAV flights, ideally at the same time of day. If the simulation results had been negative, we would not pursue such physical experiments. However, these positive results suggest that it would be valuable to perform them.

Acknowledgments

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab and the CITRIS "People and Robots" (CPAR) Initiative. The authors were supported in part by Award 2017-67021-25925, RAPID: Robot-Assisted Precision Irrigation Delivery, from the USDA under the NSF National Robotics Initiative and by donations

from Siemens, Google, Honda, Intel, Comcast, Cisco, Autodesk, Amazon Robotics, Toyota Research Institute, ABB, Samsung, Knapp, and Loccioni. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Sponsors. We thank our colleagues who provided helpful feedback and suggestions, in particular Ron Berenstein, Steve McKinley, Roy Fox, Nate Armstrong, Andrew Lee, Martin Sehr, and especially Luis Sanchez, Mimar Alsina, and the E. & J. Gallo Winery for vineyard photos and discussions.

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems", *arXiv preprint arXiv:1603.04467*, 2016.

[2] S. Ahmad, A. Kalra, and H. Stephen, "Estimating soil moisture using remote sensing data: A machine learning approach", *Advances in Water Resources*, vol. 33, no. 1, pp. 69–80, 2010.

[3] R. Bachour, W. R. Walker, A. M. Ticlavilca, M. McKee, and I. Maslova, "Estimation of spatially distributed evapotranspiration using remote sensing and a relevance vector machine", *Journal of Irrigation and Drainage Engineering*, vol. 140, no. 8, p. 04 014 029, 2014.

[4] R. Berenstein, R. Fox, S. McKinley, S. Carpin, and K. Goldberg, "Robustly adjusting indoor drip irrigation emitters with the toyota hsr robot", in *ICRA*, IEEE, 2018.

[5] R. Berenstein, A. Wallach, P. E. Moudio, P. Cuellar, and K. Goldberg, "An open-access passive modular tool changing system for mobile manipulation robots", in *CASE*, IEEE, 2018.

[6] P. Casadebaig, L. Guilioni, J. Lecoeur, A. Christophe, L. Champolivier, and P. Debaeke, "Sunflo, a model to simulate genotype-specific performance of the sunflower crop in contrasting environments", *Agricultural and forest meteorology*, vol. 151, no. 2, pp. 163–178, 2011.

[7] J. Das, G. Cross, C. Qu, A. Makineni, P. Tokekar, Y. Mulgaonkar, and V. Kumar, "Devices, systems, and methods for automated monitoring enabling precision agriculture", in *CASE*, IEEE, 2015, pp. 462–469.

[8] C. v. Diepen, J. Wolf, H. v. Keulen, and C. Rappoldt, "Wofost: A simulation model of crop production", *Soil use and management*, vol. 5, no. 1, pp. 16–24, 1989.

[9] R. Feddes, P. Kabat, P. Van Bakel, J. Bronswijk, and J. Halbertsma, "Modelling soil water dynamics in the unsaturated zone—state of the art", *Journal of Hydrology*, vol. 100, no. 1-3, pp. 69–111, 1988.

[10] J. Gago, C. Douthe, R. Coopman, P. Gallego, M. Ribas-Carbo, J. Flexas, J. Escalona, and H. Medrano, "Uavs challenge to assess water stress for sustainable agriculture", *Agricultural Water Management*, vol. 153, pp. 9–19, 2015.

[11] D. V. Gealy, S. McKinley, M. Guo, L. Miller, S. Vougioukas, J. Viers, S. Carpin, and K. Goldberg, "Date: A handheld co-robotic device for automated tuning of emitters to enable precision irrigation", in *CASE*, IEEE, 2016, pp. 922–927.

[12] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks", in *AISTATS*, 2011, pp. 315–323.

[13] L. Hassan-Esfahani, A. Torres-Rua, A. Jensen, and M. McKee, "Assessment of surface soil moisture using high-resolution multi-spectral imagery and artificial neural networks", *Remote Sensing*, vol. 7, no. 3, pp. 2627–2646, 2015.

[14] L. Hassan-Esfahani, A. Torres-Rua, A. Jensen, and M. Mckee, "Spatial root zone soil water content estimation in agricultural lands using bayesian-based artificial neural networks and high-resolution visual, nir, and thermal imagery", *Irrigation and Drainage*, vol. 66, no. 2, pp. 273–288, 2017.

[15] L. Hassan-Esfahani, A. Torres-Rua, A. M. Ticlavilca, A. Jensen, and M. McKee, "Topsoil moisture estimation for precision agriculture using unmmaned aerial vehicle multispectral imagery", in *IGARSS*, IEEE, 2014, pp. 3263–3266.

[16] S. Haug and J. Ostermann, "A crop/weed field image dataset for the evaluation of computer vision based precision agriculture tasks.", in *ECCV Workshops (4)*, 2014, pp. 105–116.

[17] C.-W. Hsu, C.-C. Chang, C.-J. Lin, *et al.*, "A practical guide to support vector classification", 2003.

[18] R. D. Jackson, S. Idso, R. Reginato, and P. Pinter, "Canopy temperature as a crop water stress indicator", *Water resources research*, vol. 17, no. 4, pp. 1133–1138, 1981.

[19] J. Jones, G. Tsuji, G. Hoogenboom, L. Hunt, P. Thornton, P. Wilkens, D. Imamura, W. Bowen, and U. Singh, "Decision support system for agrotechnology transfer: Dssat v3", in *Understanding options for agricultural production*, Springer, 1998, pp. 157–177.

[20] D. Kingma and J. Ba, "Adam: A method for stochastic optimization", *arXiv preprint arXiv:1412.6980*, 2014.

[21] N. Kumar, P. N. Belhumeur, A. Biswas, D. W. Jacobs, W. J. Kress, I. C. Lopez, and J. V. Soares, "Leafsnap: A computer vision system for automatic plant species identification", in *Computer Vision–ECCV*, Springer, 2012, pp. 502–516.

[22] Y. Lin, M. Kang, and J. Hua, "Fitting a functional structural plant model based on global sensitivity analysis", in *CASE*, IEEE, 2012, pp. 790–795.

[23] L. Mateos, J. Berengena, F. Orgaz, J. Diz, and E. Fereres, "A comparison between drip and furrow irrigation in cotton at two levels of water supply", *Agricultural water management*, vol. 19, no. 4, pp. 313–324, 1991.

[24] A. McBratney, B. Whelan, T. Ancev, and J. Bouma, "Future directions of precision agriculture", *Precision agriculture*, vol. 6, no. 1, pp. 7–23, 2005.

[25] M. Möller, V. Alchanatis, Y. Cohen, M. Meron, J. Tsipris, A. Naor, V. Ostrovsky, M. Sprintsin, and S. Cohen, "Use of thermal and visible imagery for estimating crop water status of irrigated grapevine", *Journal of experimental botany*, vol. 58, no. 4, pp. 827–838, 2007.

[26] C. Nendel, "Monica: A simulation model for nitrogen and carbon dynamics in agro-ecosystems", in *Novel Measurement and Assessment Tools for Monitoring and Management of Land and Water Resources in Agricultural Landscapes of Central Asia*, Springer, 2014, pp. 389–405.

[27] Z. Qiu, J. Mao, and Y. He, "The design of field information detection system based on microcomputer", in *CASE*, IEEE, 2006, pp. 156–160.

[28] G. H. Schmitz, N. Schütze, and U. Petersohn, "New strategy for optimizing water application under trickle irrigation", *Journal of irrigation and drainage engineering*, vol. 128, no. 5, pp. 287–297, 2002.

[29] O. Söderkvist, *Computer vision classification of leaves from swedish trees*, 2001.

[30] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting.", *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[31] T. C. Thayer, S. Vougioukas, K. Goldberg, and S. Carpin, "Multi-robot routing algorithms for robots operating in vineyards", in *CASE*, IEEE, 2018.

[32] ——, "Routing algorithms for robot assisted precision irrigation", in *ICRA*, IEEE, 2018.

[33] J. Van Dam, J. Huygen, J. Wesseling, R. Feddes, P. Kabat, P. Van Walsum, P. Groenendijk, and C. Van Diepen, "Theory of swap version 2.0; simulation of water flow, solute transport and plant growth in the soil-water-atmosphere-plant environment", DLO Winand Staring Centre [etc.], Tech. Rep., 1997.

[34] J. S. Wallace and P. J. Gregory, "Water resources and their use in food production systems", *Aquatic Sciences-Research Across Boundaries*, vol. 64, no. 4, pp. 363–375, 2002.

[35] J. Wallace, "Increasing agricultural water use efficiency to meet future food production", *Agriculture, ecosystems & environment*, vol. 82, no. 1, pp. 105–119, 2000.

[36] S. G. Wu, F. S. Bao, E. Y. Xu, Y.-X. Wang, Y.-F. Chang, and Q.-L. Xiang, "A leaf recognition algorithm for plant classification using probabilistic neural network", in *ISSPIT*, IEEE, 2007, pp. 11–16.

[37] H. Xiang and L. Tian, "Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (uav)", *Biosystems engineering*, vol. 108, no. 2, pp. 174–190, 2011.

[38] Y. Yang, D. Fleisher, D. Timlin, B. Quebedeaux, and V. Reddy, "Evaluating the maizsim model in simulating potential corn growth.", in *The 2008 Joint Annual Meeting*.

[39] B. Zaman, M. McKee, and C. M. Neale, "Fusion of remotely sensed data for soil moisture estimation using relevance vector and support vector machines", *International journal of remote sensing*, vol. 33, no. 20, pp. 6516–6552, 2012.