CSCI 631 : Foundations of Computer Vision Project
Group Members : Divyank Kulshrestha , Vineet Singh

Abstract

The objective of the project is to develop an algorithm to classify images of cotton crops based on its irrigation treatment. We have four possible irrigation treatments – Rainfed, Fully Irrigated, Percent Deficit and Time Delay. In easy words, we have to look at images of cotton crops and tell what type of irrigation it grew under. The first step is to read in and process the TIF images using processes such as sub-sampling, separating the crop from background, removing noise and cleaning image using morphological operations. After that, we extracted the features from the processed images on which the decision tree will be trained. We split the provided data into different ratios – used for training and for testing. Some of the features we chose for training were the intensity of green crop color, number of cotton buds and the area covered by the crop . Finally, once the model is trained, we make predictions using the trained model on the testing data. Our model returns an accuracy of about 82-86%.

Introduction

The US is a major producer and exporter of cotton, but frequent droughts and declining groundwater levels have led to a decrease in cotton production in the southern states, known as the Cotton Belt. To ensure long-term stability and profitability, water resources must be used sustainably, and crop water use efficiency must be maximized to reduce water requirements in regions with water scarcity. However, current irrigation scheduling methods may not accurately estimate crop water requirements. The goal of this project is to develop an algorithm using computer vision techniques to classify the irrigation treatment of a cotton image. The image is assumed to have the cotton plant in the center and will be one of the four irrigation treatments: Rainfed, Fully irrigated, Percent deficit, or Time delay. This project requires object segmentation and image classification skills. Top-view images of cotton crops, taken using a DJI Phantom 4 drone, and labels for those images are provided. A part of this data is to be used to train a decision tree model. Training involves processing and extracting useful features from the images on which decisions can be made. Using the same model, we predict if a cotton crop belongs to the rainfed, fully irrigated, percent deficit or time delay category by looking at its top-view image.

Methodology

In this project, our main goal was to examine cotton crop top-view photos and extract pertinent features that can be used to train a machine learning classifier. Our primary objectives were to locate the green area (which represents the healthy crop region), count the number of white cotton buds (a crucial indicator of the cotton yield), and calculate the mean green intensity (which offers valuable information about the crop's overall health). To achieve this, we created a thorough MATLAB program that processes an entire set of images, extracts the desired features, and subsequently trains a decision tree classifier to predict labels for the images based on these extracted features.

Data Preparation and Iteration Process

The initial part of the code is responsible for initializing the appropriate variables and setting up a loop to cycle over all of the accessible images. The images are stored in a directory called cotton images (this was provided), and their corresponding labels can be found in a CSV file called labels_rgb.csv (this was also provided). We load these text labels into a categorical array. The loop iterates through all the images, invoking a function called CV_test_GreenMask.This function does the following first step is to read in the images and prepare them for pre-processing. The images provided were in TIFF format and had 4 channels instead of the usual 3. We had to isolate the 3 RGB channels before further processing of the image, then it extracts the features, and appends the extracted features to a feature matrix X and the corresponding label to a label vector Y. A regex expression was used to compute the name and location of images to be read during the data prep phase.

Original image (Type Rainfed)

Feature Extraction: Green Mask Refinement

We first read the input image and separate the red, green, and blue color channels within the CV_test_GreenMask function. Then, in order to create a binary green mask based on these thresholds, we define color channel thresholds for green (redThres, greenThres, and blueThres). The areas of the image that are primarily green in color—which correlate to healthy cotton crops—are represented by this mask.

We use morphological techniques like dilatation and hole-filling to further refine the green mask. A disk-shaped structural element (se) with a predetermined radius is used to execute dilation; its radius can be changed depending on the specific image properties (after multiple iterations we found that size=3 is best for out usecase). This operation is beneficial for filling small gaps and connecting fragmented green regions. After dilation, we utilize the **imfill** function with the 'holes' option to fill any remaining holes in the dilated green mask. This refined mask, referred to as filledGreenMask, represents a more accurate and complete representation of the green patch in the image.

Green mask Applied

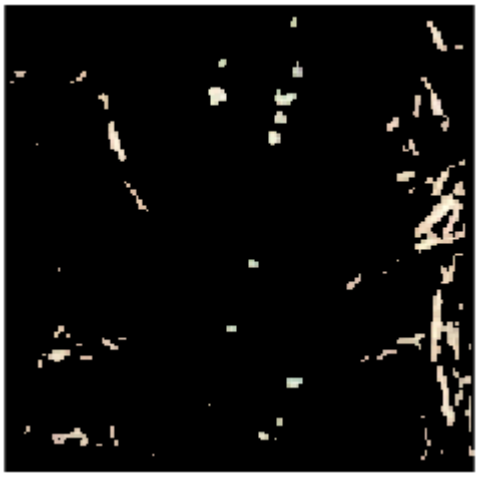
Post dilation and filling

Feature Extraction: Identifying White Cotton Buds

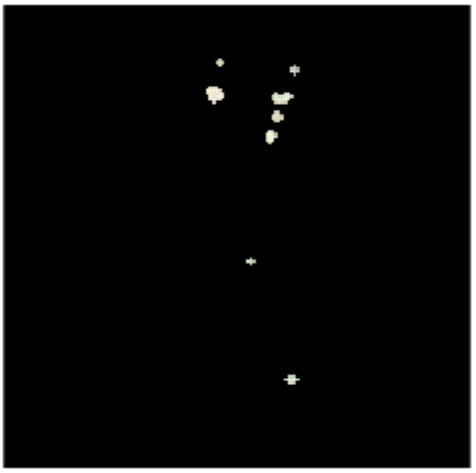
Subsequently, using a different set of color channel thresholds (redThresWhite, greenThresWhite, and blueThresWhite), we produced a binary white mask. This mask locates areas of the image that are primarily white in color, which are cotton buds. Using the logical & operator , we combined the white mask with the filled green mask to create a mask (whiteObjectsOnGreenPatch) that represents the white objects (cotton buds) on the green patch.

We then used a morphological opening procedure on the white objects mask to remove noise and small white items. A disk-shaped structuring element (seOpening) with a predetermined radius that can be changed depending on the properties of the image is used to execute the opening operation. This procedure is useful for getting rid of little white items that might not actually be cotton.

Once the white object's mask has been cleaned, we labeled the connected components (white cotton buds) in the mask using the **bwlabel** function. This function returns a labeled matrix (labeledWhiteObjects) and the number of connected components found (numWhiteCottonBuds), which represents the number of white cotton buds in the image.



White mask applied



Post noise removal & cleanup

Feature Extraction: Green Area and Mean Green Intensity Calculation

We determined the area of the green patch by counting the non-zero pixels in the filled green mask (greenArea). This value represents the total number of pixels in the green patch, which can be used as an indicator of the size of the healthy crop region.

To compute the mean green intensity, we extracted the green channel values within the green patch using logical indexing with the filled green mask (greenIntensityInPatch). We calculated the mean value of these green intensity values using the mean function with the 'all' option (meanGreenIntensity). This value represents the average green color intensity within the green patch and can be used as an indicator of the overall health of the cotton crops in the image.

Feature Vector Construction and Classifier Training Process

Once the features were extracted from all the images, the program created a feature vector by combining the calculated green area, the number of white cotton buds, and the mean green intensity for each image. This was then appended to the feature matrix X, and the corresponding label was appended to the label vector Y.

After processing all the images, the program trained a decision tree classifier using the ‘fitctree’ function. To assess the classifier's performance, we split the data into training and testing sets using a 70:30 split. The **cvpartition** function created a partitioned dataset with the specified 'HoldOut' percentage, which was used to divide the feature matrix X and label vector Y into training (Xtrain, Ytrain) and testing (Xtest, Ytest) datasets.

The decision tree classifier was trained on the training set (Xtrain, Ytrain) and its performance was evaluated on the test set (Xtest, Ytest). The classifier's predictions on the testing dataset were stored in the Ypred vector. The accuracy of the classifier was calculated as the percentage of correctly predicted labels in the testing dataset, by dividing the sum of matching elements between Ypred and Ytest by the total number of elements in Ytest. The calculated accuracy was then displayed as output.

In summary, we developed a MATLAB program that processes top-view images of cotton crops, extracts features related to the green area, white cotton buds, and mean green intensity, and trains a decision tree classifier to predict labels for the images based on these features. This approach can be useful for agricultural applications, such as monitoring crop health, estimating crop yield, and making informed decisions about crop management. By analyzing the images and extracting relevant features, we can obtain valuable insights into the current state of the cotton crops and potentially improve agricultural practices. This project demonstrates the power of image processing and machine learning techniques in providing useful information for better decision-making.

Results Discussion (what worked and did not work)

Solution that worked for us:

- The objective of this project was to create a machine learning classifier that could analyze top-view photographs of cotton fields, extract useful attributes, and classify the crops as well irrigated or not. Green area, white cotton bud count, and mean green intensity were the variables of interest because they are all indicators of the general irrigation pattern of the crop.
- For our solution, we developed a MATLAB program that could read in a directory of cotton crop photos along with the labels in a CSV file for each image. It then went over each image iteratively, processed them, and finally extracted the key attributes.

- Using a green color channel threshold, a binary green mask was made to separate the crop from the background. This was followed by morphological procedures like dilation and hole-filling. The white cotton buds were also identified using a binary white mask made using a different channel threshold, and the noise was subsequently removed using morphological opening.

- The program extracted the green channel values within the green patch using logical indexing with the filled green mask, computed the green area by adding the pixel values in the filled green mask, and determined the mean green intensity. Using the ‘bwlabel’ function to label the connected components of the white mask, the quantity of white cotton buds was calculated.

- The program then used the computed green area, white cotton bud count, and mean green intensity for each image after processing each one as training features. The decision tree classifier was trained using these features and the ‘fitctree' function, and it was then tested on a test dataset. By dividing the total number of elements in the test dataset by the sum of predictions matching with actual labels, the accuracy of the classifier was determined.

Overall, the solution involved a thorough image processing and machine learning approach that successfully classified the top-view images of cotton crops into different categories based on the relevant features extracted from

the images. For a 70:30 data split, our model predicts with an accuracy between the range of 83% to 86%. Table below shows the accuracy achieved for different splits of data into training and testing datasets.

Training Data Proportion	Testing Data Proportion	Accuracy
60%	40%	81 - 84%
70%	30%	83 - 86%
80%	20%	84 - 87%
90%	10%	81 - 86%

A few of the things we tried but unfortunately did not work:

- In order to make computations faster, we tried converting the image to double using **im2double** function. However, that slashed the accuracy by 50% and thus we had to avoid the conversion to double. This conversion breaks the algorithm.

- We tried to isolate and use the red channel for further processing. Although the red channel provided the best contrast between the crop and the background in comparison to green or blue channels, we could only achieve an accuracy of about 60%. Hence, we had to look for alternative solutions.

- We tried to convert the image to grayscale and binarize it using the **imbinarize** function. To clean that binary image, we performed morphological operations such as erosion, opening and dilation. However, it was hard to balance the cleaning vs accuracy metric and could only achieve about 60% accuracy. Thus, this solution was also scrapped.

- We tried to count white pixels and use it as a feature to account for cotton on plants, but that did not work as there was negligible change in accuracy. Some of the buds seemed to be green-ish which made it hard to detect all the buds accurately.

Conclusion

This project was a really interesting one. Initially, it looked like a very difficult task to accomplish, but as we read into it more, it seemed easier and easier. The first task that was new for us was working with a TIFF image. We realized that the images actually had 4 channels, which made things such as double conversion complicated and we had to isolate useful channels. After trying out some morphological processing such as binarizing and image cleaning and utilizing features that we obtained from 'imageprops', we realized that we were stuck at ~60% of accuracy, and no matter what modifications we tried, that number would not change.

We finally scrapped all that we had done till then and started from scratch with fresh eyes. We looked at a bunch of images side-by-side, and came up with new features:

- Intensity of green in the crop
- Area covered by the crop, which is basically the area of green color in the image
- Number of cotton buds in the image

To isolate the green crop and cotton buds, we used two image masks of green and white color separately. One tricky part to deal with was detecting the cotton buds. It seemed that in some images, cotton buds had a green tint in their color, instead of being completely white. Finally, we combined the green and white masks and applied it to obtain images that only had the green crop and cotton buds in it. All other pixels which made up the background or noise were set to zero.

To calculate the intensity of the green crop, we calculated the mean of intensities of green color in the crop and used that as a feature. To count the cotton buds, we found the number of connected components from the image obtained by applying the combined masks on the image using the 'bwlabel' function. Area of green was easy to calculate by applying the green mask and summing the green pixels that remained.

We trained our decision tree using these three features described above, and finally achieved the desired accuracy. Our model predicts with an accuracy between the range of 82% to 86%, with the best and most consistent accuracy achieved for an 80:20 data split, which was mostly around 85% and sometimes even touching 87%.