Lahore University of Management Sciences
Syed Babar Ali School of Science and Engineering, Department of Electrical Engineering
EE100 – Engineering Laboratory

Fall 2021

Project 1 – Handout

Machine Vision in Agricultural Robots

1. OBJECTIVE

To detect crop lanes and individual plants for spraying, weeding, harvesting or health monitoring using a camera mounted on a mobile robot.

2. BACKGROUND

According to some estimates, the agricultural robots market is expected to exceed $10 billion by 2025. Emerging applications of robots or drones in agriculture include weed control, seeding, harvesting, health monitoring, fruit counting, soil analysis and pest-attack remediation. Agricultural applications are the forefront of robotics research because an agricultural field is one of the most challenging environments for inspection, locomotion and intervention. Robotics is attractive to farming businesses as it promises lower production costs, higher quality of produce and a decreased need for manual labor. With the advent of electrical vehicles and aerial drones, small robots offer many advantages over bulky human operated vehicles such as tractors which have a negative impact on soil and the environment.

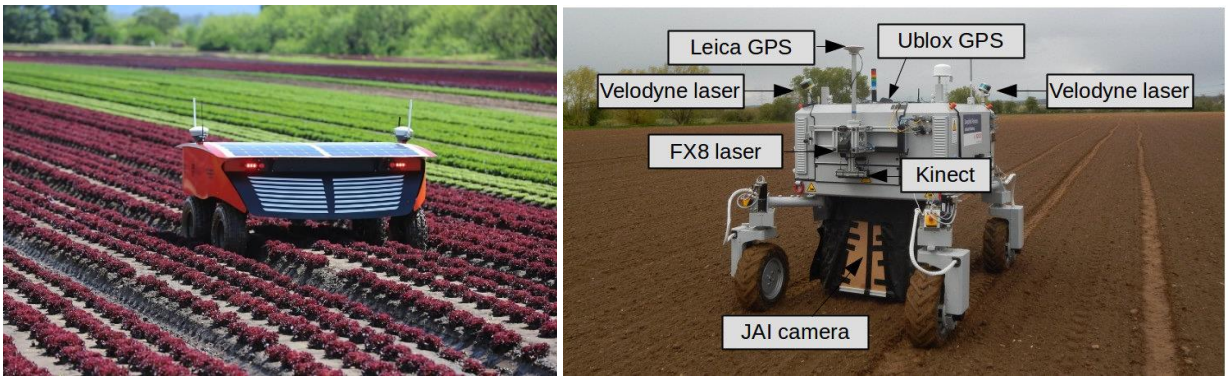

Figure 1. Agricultural robots developed by Australian Center for Field Robotics (ACFR), Universiy of Sydney [Left] and Bosch Deepfield Robotics. [Right]

Like most robots with a high degree of autonomy, one of the most critical components of an agricultural robot is its perception. Machine vision systems, enabled by high-performance cameras, laser scanners and obstacle detection sensors help the robots to navigate in rough terrain, build maps of the environment, detect objects of interest (e.g. fruits, leaves and plants) and deliver precisely controlled inputs where needed. Due to the availability of computing resources, memory, storage and advancements in algorithms, machine-vision enabled robotic systems are increasingly becoming sophisticated in guidance and inspection tasks related to various operations on the agricultural field.

3. PROBLEM SETUP

In this project, you will be asked to analyze images collected from a downward looking high-resolution RGB camera mounted on a mobile robot. The camera points to the ground and is meant to inspect corn plants at various stages of their development. The crop is usually seeded in straight lines in an open filed. The robot has been programmed to move forward linearly, thereby keeping two lines of the crop in the camera's field of view. The images are stored on an on-board computer and retrieved for later analysis. GPS, clocks and other localization & timing sensors help tag the collected imagery with events and locations on the field.

Figure 2. An agricultural robot moving over rows of plants [Left]. Image captured from a camera mounted underneath the robot with some artificial illumination [Right]. (Data and images courtesy of ACFR.)

4. DATA

Imagery data to be used for the project is a small video captured from a robot camera. The video has been given to you as a series of consecutive images in a folder titled `Images` on LMS. Each file has a name that indicates its date and time of capture. E.g. `20141125T061716.838366.png`, is an image representing a video frame captured on Nov 25th, 2014 with a time-stamp given by the number 061716.838366. Sorting the filename in alphabetical order (Perform 'Sort by Name' in Windows) will give you the correct chronological order of the files. Note in particular that the time-stamp numbers increase monotonically and provides a natural ordering of the images.

Preparing for batch-processing of image files

The following template can help you set up a data processing pipeline to process all the images in one go. This will help you in cutting down the time and effort needed in loading each image file manually into your code. Note that for a very large number of images (or files), this is an unavoidable step.

Create a folder by the name of images in your MATLAB working directory and copy all files into it. The following code demonstrates how is it possible to load each file automatically, do a simple image processing step and display the results in a batch.

Sample code is given below.

```
clear; close all; clc % clear workspace at the beginning

dataPath = 'images/'; % a string for the path to folder where image are placed

imagesList = dir([dataPath '*.png']); % Get a list of all PNG files in the path

imgNo = 1;    % Let's process first image in the directory, as an example

im_raw = imread([dataPath imagesList(imgNo).name]); % Load first file as MATLAB
array

im_process = im2double(im_raw); % A rescaling function from Image Processing
toolbox

% We can process other files in a similar way

noImg = length(imagesList); % Number of files in folder

for indx = 1:noImg

    im_raw = imread([dataPath imagesList(indx).name]);

    im_processed = im2double(im_raw);

    im_processed2 = flipud(im_processed);

    subplot(1,2,1); imagesc(im_raw);

    title(sprintf('Original Image %i',indx)); axis image

    subplot(1,2,2); imagesc(im_processed2);

    title('Result'); axis image

    pause(1); % pause for 1 sec before showing next results

end
```

For a screenshot of results see Figure 3.

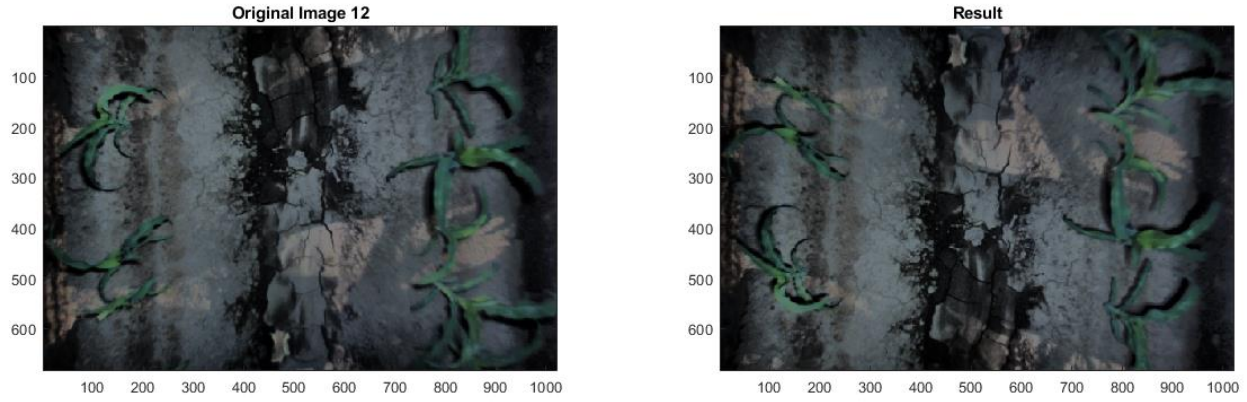

Figure 3. A screenshot from batch processing. The result is a flipped upside-down image of the 12th image file.

5. PROJECT TASKS

You can also see the accompanying video for further clarification of the tasks if desired. **Note that upon completion of the tasks, you must submit your MATLAB code and a report describing all your steps and rationale for any design choices that were made in the process.**

Task 1: Pre-processing the Images

In this task you will perform the histogram equalization and brightness adjustment using tools similar to those introduced in Lab 7. Play with the following two steps to get images of a good contrast and brightness.

1. Write a MATLAB code for histogram equalization for each image. As an intermediate step, convert the images from RGB to HSV color space. HSV stands for Hue, Saturation, Value.
2. If required, do a brightness adjustment by amplifying each pixel by a factor of 2 in each channel. If any pixels are excessively bright, clip them to a max level of 1.

Hint. Explore MATLAB functions `rgb2hsv()`, `hsv2rgb()` and `histeq()`. You may find it advantageous to to perform histogram equalization in the V channel of an HSV image. Your results may look similar to the screenshots in Figure 4.

Task 2: Compute a Vegetation Index to Segment Chlorophyll from Soil

Following the pre-processing steps of Task 1, you are now ready to extract useful information related to agriculture. Multi-spectral images (such as normal RGB images) can be used to detect vegetation using a variety of indices that have been developed by various researchers. The common observation in the development of these indices is that vegetation (chlorophyll), soil, water and other elements of interests in an agricultural field absorb and reflect radiations of various frequencies differently. Based on these differences, one can distinguish the nature of the material. In Figure 5, the absorption spectrum of soil, vegetation and water have been given.

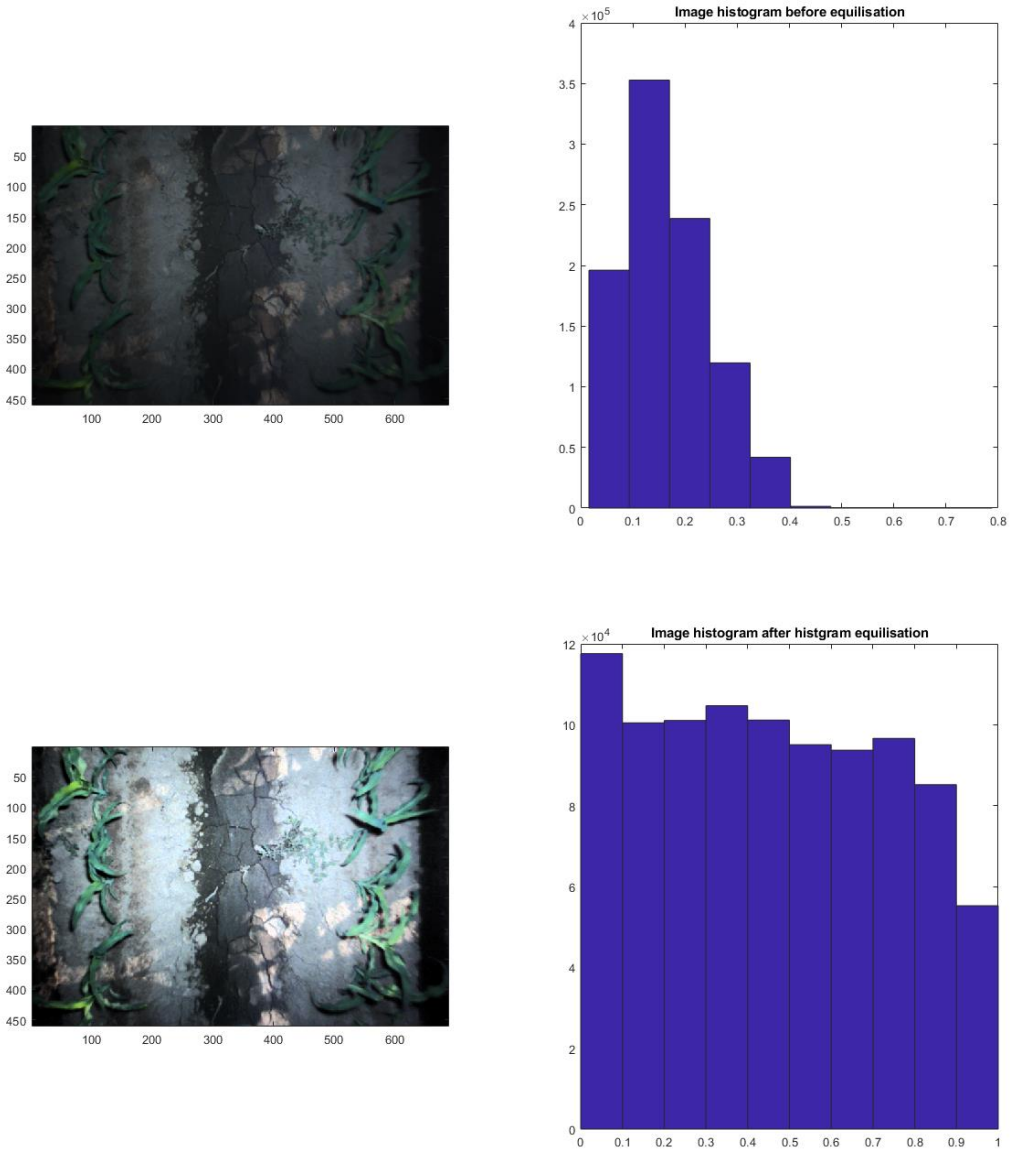

Figure 4. Results of histogram equalization. Original image captured under bad illumination conditions and the result of histogram equalization.

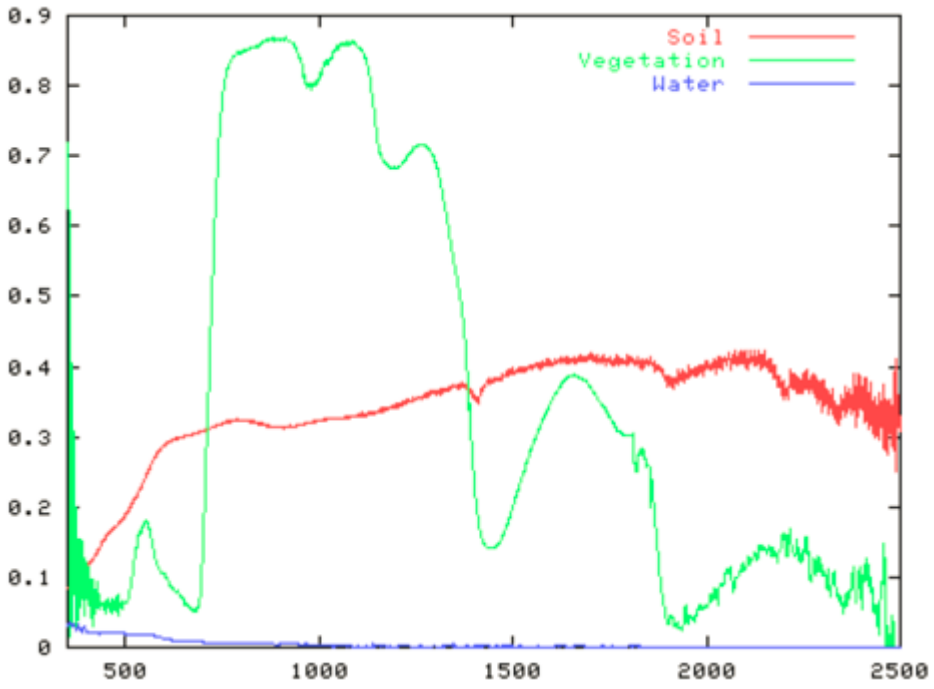

Figure 5. Spectral responses of various materials at various wavelengths (nm).

Some popular Vegetation indices in the RGB space are:

Normalized Difference Index (NDI) $= \frac{G - R}{G + R}$

Excess Green (ExG) = 2G – R – B.

Excess Red (ExR) = 1.4R – G.

Excess Blue (ExB) = 1.4B – G.

Excess Green minus Excess Red (ExGR) = ExG – ExR - ExB

1. Use the pre-processed images in the RGB color space to compute the NDI and ExGR indices. It may be helpful to make separate MATLAB functions for each index since they will be called repeatedly in later tasks. Store the computation against each pixel in a separate variable and show the results as a batch process on all images.
2. Next, prepare a mask for the removal of background. Use a threshold on the computed index (ExGR or NDI) at each pixel to create a binary image (0 or 1). Adjust the thredhold to capture maximal portions of the image that pertain to vegetation. We now have a mask. Use this mask to set all the non-vegetation portions of the images to zero. This process is called segmentation.

See Figure 6 for expected results of this task. Do a batch processing of this task on all images.

Hint.

1. ExGR works reasonably well for a fixed threshold of zero. (It has been designed as a zero cross-over detector).
2. Manually adjusting the threshold for each image may be difficult for other indices. You will see that with a fixed threshold across all images, vegetation in some images will be excessively missed and in some large portions of the background will remain. Explore the MATLAB function `graythresh()` from the Image Processing Toolbox that computes the "best" threshold automatically.

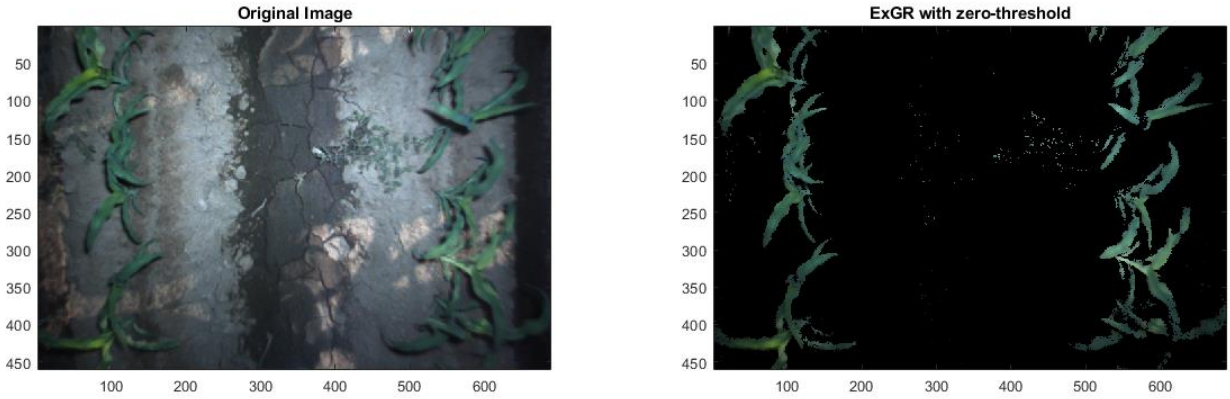

Figure 6. Results of masking the original image with a thresholded ExGR output.

Task 3: Crop Lane Detection

After obtaining good segmentation results from Task 2, we are ready to determine crop lanes automatically. If we are successful, this information can be used to correct the course of the robot so that it moves forward without damaging the plants. Moreover, the various agricultural implements such as spray nozzles can be positioned optimally for maximum impact.

1. Use the results of the masked images in the next steps. Collapse the NxM image matrix onto a row vector in which each bin counts the number of green pixels in each column of the masked image. Graph the vector right beneath the image and observe how the vertical concentration of green pixels in the image correspond to high values of this graph. Typically, you will see two peaks, each corresponding to a crop lane. Typical results are depicted in Figure 6.
2. Write a code to detect the two peaks using your favorite method of peak detection. Overlay the positions of the two lanes on your original image to study how robust your method is.
3. Since the peak detection method is expected to give you large horizontal jump, write a filtering routine on each of the detected lane's horizontal positon to make the results stable. You are welcome to use any method of your choice that provides a smooth estimation of the lanes, which could be used by a nozzle spray etc.
4. We are slightly lucky here, because the motion of the robot is already aligned with the lane direction. Imagine that the robot was moving at an angle (less than 45 degrees) with respect to the crop lanes. How would you modify the algorithm above to detect the crop lines?

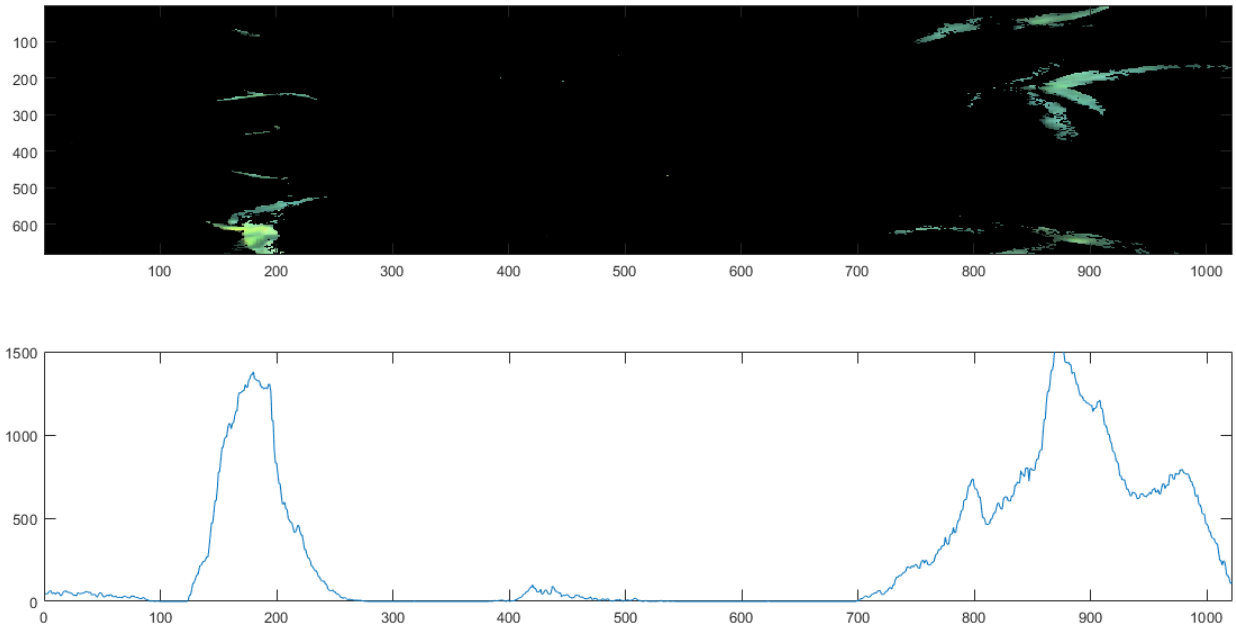

Figure 6. Expected results of Task 3.1.

Task 4: Open Ended Task – Count the Number of Plants in the Entire Sequence

Develop and test an algorthim to count the number of unique plants in the entire sequence of images. Please keep in mind that successive frames may be showing the same plant as it appears and disappears. Be sure that you are able to justify your count using good visualization and reporting.

CHECKPOINTS AND TIMELINE

S.No	Tasks	Deadline	Instructions
1	Project selection and team formation	**Saturday, November 27, 2021 by 11:55 pm**	Send an email to Nouman Arshad (nouman.arshad@lums.edu.pk) indicating your choice of project and a list of team members. **Max. No. of Team Members:** 3 **Min. No. of Team Members:** 2
2	**Checkpoint 1:** Complete Task 1&2	**Thursday, December 2, 2021 by 6:00 pm**	You will be assigned a dedicated TA who will track your progress. Show your code and results to the TA in the lab/office hours.
3	**Checkpoint 2:** Complete Task 3 & 4	**Thursday, December 9, 2021 by 6:00 pm**	Show your code and results to your TA in the lab/office hours.
4	Final code and report submission with all 4 tasks completed	**Friday, December 10, 2021 by 11:55 pm**	Submit your MATLAB code and any stipulated documentation.