University of Nebraska - Lincoln

DigitalCommons@University of Nebraska - Lincoln

Mechanical (and Materials) Engineering --
Dissertations, Theses, and Student Research

Mechanical & Materials Engineering, Department
of

Spring 4-22-2016

Automated Mini-Channel Platform For Studying Plant Root Environments

Kevin F. Kreis

University of Nebraska-Lincoln, kevin.kreis@gmail.com

Follow this and additional works at: <http://digitalcommons.unl.edu/mechengdiss>

Part of the [Agriculture Commons](#), [Mechanical Engineering Commons](#), and the [Plant Sciences Commons](#)

Kreis, Kevin F, "Automated Mini-Channel Platform For Studying Plant Root Environments" (2016). *Mechanical (and Materials) Engineering -- Dissertations, Theses, and Student Research*. 98.
<http://digitalcommons.unl.edu/mechengdiss/98>

This Article is brought to you for free and open access by the Mechanical & Materials Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in Mechanical (and Materials) Engineering -- Dissertations, Theses, and Student Research by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.

AUTOMATED MINI-CHANNEL PLATFORM FOR
STUDYING PLANT ROOT ENVIRONMENTS

by

Kevin Frank Kreis

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfillment of Requirements

For the Degree of Master of Science

Major: Mechanical Engineering and Applied Mechanics

Under the supervision of Professor Sangjin Ryu

Lincoln, Nebraska

April 2016

AUTOMATED MINI-CHANNEL PLATFORM FOR STUDYING PLANT ROOT ENVIRONMENTS

Kevin Frank Kreis, M.S.

University of Nebraska, 2016

Advisor: Sangjin Ryu

Plants are crucial to our lives; they provide us with building materials, oxygen, and food. A season's crop yield can be significantly affected by local environmental factors. Farming practices currently focus on using fertilizer, pesticides, monitoring water availability, and genetic modification of the plant to increase crop yield. Improving fundamental understanding of plant root interactions with their local soil environment, or rhizosphere, will help improve crop yield. Studying such interactions is challenging because roots are underground, making it difficult to observe interactions and to manipulate the local soil environment.

The goal of this thesis is to develop an automated mini-channel platform to investigate how plant roots respond to changes in their environment. Corn seedlings were grown inside the transparent mini-channel device. The automated system maintains the level of growth medium in the device to ensure the plant stays hydrated. A digital camera regularly images the root growing in the device. The images are processed to characterize the root's growth. The device accommodates electrochemical sensors to measure changes in nitrate concentration.

The automated platform was developed to simplify researching plant root-environment interactions, with the goal of improving crop yields. It measured corn's

growth rate over time, and determined that the con consumed nitrate over time. The platform's adaptable design, simple fabrication, and low cost make it simple to replicate and use to study different plants and environmental stimuli. Improving our understanding of different plant root-environment interactions will be crucial in improving crop yields to match expected population growth.

ACKNOWLEDGMENTS

I would like to thank my family, friends, and most importantly my wife, Dani, for their unwavering support. I would like to thank Dr. Sangjin Ryu for supporting and enabling my research. I'd also like to thank Dr. Rebecca Lai and Hamid R. Lotfi Z.Z. of the Lai Lab at UNL for their work incorporating their nitrogen detecting electrode into my platform. I'd like to thank Dr. Aaron Lorenz for supplying the corn seeds. I'd like to thank Dr. Daniel Schachtman for his assistance and support. I would also like to thank NASA Nebraska Space Grant for the generous fellowship. Finally, I would like to thank Drs. Kevin Cole and Benjamin Terry for being in my thesis committee.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	iii
LIST OF MULTIMEDIA OBJECTS	vii
LIST OF TABLES	x
CHAPTER 1. INTRODUCTION	1
1.1. Literature Review: Plant Root-Environment Interactions.....	1
1.2. Literature Review: Microfluidic Study of Plants	4
1.2.1. RootChip.....	5
1.2.2. Plant Chip	6
1.2.3. Chemical Stimulation of Discrete Root Sections	7
1.3. Experimental Objectives and Goals	7
CHAPTER 2. METHOD: ROOT GROWTH CHARACTERIZATION	9
2.1. MATLAB Image Analysis	9
2.1.1. Row Progression: Approximate Method	11
2.1.2. Euclidean Distance Transform: Exact Method.....	12
2.1.3. Validation	13
2.2. Control Group	15
2.2.1. Test Procedure	15
2.2.2. Control Test Results	17

CHAPTER 3. METHOD: MINI-CHANNEL DEVICE.....	18
3.1. Version 1: Prototype Device	18
3.1.1. Mini-Channel Device Goals and Design	18
3.1.2. Fabrication and Design Development	21
3.1.3. Experimental Results	23
3.2. Version 2: Dual Glass Plate	24
3.2.1. Design	24
3.2.2. Fabrication and Design Development	26
3.2.3. Experimental Results	26
3.3. Version 3: Electrodes	28
3.3.1. Design	28
3.3.2. Fabrication and Design Development	29
3.3.3. Experimental Results	30
CHAPTER 4. METHOD: PLATFORM DEVELOPMENT	33
4.1. Platform Design and Function	33
4.1.1. Growth Medium Management.....	35
4.1.2. Root Imaging	36
4.1.3. Ambient Environment Measurements	37
4.1.4. User Control, Device Feedback and Data Management.....	37
4.2. Experimental Results.....	38

4.3. Future Platform Improvements	40
CHAPTER 5. CONCLUSION.....	42
APPENDIX.A SUPPLEMENTAL FIGURES	44
APPENDIX.B MATLAB Image Analysis Validation	49
APPENDIX.C MATLAB Code	52
C.a MATLAB Automation Main Driver Program.....	52
C.b LED Color Function	64
C.c Image Analysis Function	65
C.d Camera Setup Function.....	70
REFERENCES	71

LIST OF MULTIMEDIA OBJECTS

FIGURE 1-1: PLANT ROOTS EXUDE CHEMICALS TO MEDIATE THEIR INTERACTIONS WITH MICROORGANISM IN THE RHIZOSPHERE .	3
FIGURE 1-2: ROOTCHIP ENABLES PARALLEL STUDY OF PLANT CELLULAR DEVELOPMENT WITH ENVIRONMENTAL CONTROL FOR INDIVIDUAL PLANT ROOTS. A) PLANTS GROWING IN PARALLEL WITH CONTROL AND SUPPLY MICROCHANNELS. B) ROOTCHIP SCHEMATIC SHOWS ROOT GROWING DOWN THE CURVED PLASTIC TIP INTO THE MAIN CHANNEL FOR OBSERVATION.	5
FIGURE 1-3: PLANT CHIP A) DEVICE SCHEMATIC SHOWING PARALLEL DESIGN. B) FLOW SIMULATION SHOWING AUTOMATIC SEED TRAPPING. C) EXPERIMENTAL VERIFICATION OF SEED TRAPPING.....	6
FIGURE 1-4: CHEMICAL STIMULATION OVER SMALL ROOT SEGMENTS A) LAMINAR FLUID FLOW IS USED SMALL REGIONS OF THE ROOT. B) FLORESCENT MICROSPHERES DEMONSTRATE FLOW ACROSS A TARGETED REGION OF THE ROOT. (SCALE BAR IS 200 μm)	7
FIGURE 2-1: MATLAB IMAGE PROCESSING STEPS.....	10
FIGURE 2-2: ROW PROGRESSION METHOD CANNOT ACCURATELY PROCESS SAMPLES WITH HIGH CURVATURE.	12
FIGURE 2-3: DEFORMED WIRE USED TO VALIDATE MATLAB CODE PERFORMANCE.	14
FIGURE 2-4: CORN WAS GROWN ON GERMINATION PAPER FOR THE CONTROL TEST	16
FIGURE 2-5: CONTROL TEST ROOT GROWTH. CaCl_2 AND $\text{CaCl}_2+\text{NO}_3$ GROWTH MEDIUMS ARE COMPARED.	17
FIGURE 3-1: RENDING OF PROTOTYPE MINI-CHANNEL DEVICE. PDMS BONDED TO GLASS, WHICH IS SHOWN BLACK FOR IMPROVED VISIBILITY.....	19
FIGURE 3-2: DIMENSIONED TOP DOWN VIEW OF THE PDMS MOLD.....	20
FIGURE 3-3: ISOMETRIC VIEW OF MINI-CHANNEL PDMS MOLD	20
FIGURE 3-4: PROTOTYPE 3D PRINTED MOLD	21
FIGURE 3-5: PROTOTYPE DEVICE GROWING CORN ROOT OVER 5 DAYS.	24
FIGURE 3-6: TWO GLASS PLATES SANDWICH THE DEVICE'S PDMS BODY BETWEEN THEM, PROVIDING IMPROVED CLARITY AND DURABILITY.	25
FIGURE 3-7: DUAL PLATE 3D PRINTED PDMS MOLD	26
FIGURE 3-8: COMPLETED DUAL PLATE DEVICE.....	26

FIGURE 3-9: DUAL PLATE DEVICE GROWING CORN OVER THE COURSE OF 4 DAYS.	27
FIGURE 3-10: CORN GROWN THE DUAL GLASS PLATE DEVICES HAS A GROWTH RATES CONSISTENT WITH THE CONTROL EXPERIMENTS.	28
FIGURE 3-11: VERSION 3 ELECTRODE PDMS MOLD. THE MOLD BODY (GREEN) GIVES THE PDMS ITS SHAPE, AND HAS THREE SMALL SOCKETS TO RECEIVE EMBEDDED ELECTRODES. THE IS A LID (BLUE), WITH AN EMBEDDED GASKET. THE RED PIECE HOLDS THE ELECTRODES IN POSITION WHILE THE PDMS CURES. WHILE CURING IN THE OVEN THE RED PIECE FACES UPWARD, TO PREVENT PDMS FROM SPILLING OUT AROUND THE WIRES.	29
FIGURE 3-12: ALUMINUM MOLD A) THE MOLD SHOWING PDMS WITH EMBEDDED ELECTRODES, BEFORE REMOVAL FROM THE MOLD. DEMONSTRATING HOW THE MOLD IS USED. B) A CROSS SECTION VIEW OF THE MOLD SHOWS HOW THE PLASTIC CHANNEL FITS INTO A GROOVE IN THE METAL MOLD. IT ALSO SHOWS HOW THE GASKETS WIDEN TO ENSURE THEY ARE NOT PULLED OUT OF THE MOLD.	30
FIGURE 3-13: CALIBRATING THE NITRATE CONCENTRATION MEASURING ELECTRODES A) SOLUTIONS CONTAINING DIFFERENT KNOWN NITRATE CONCENTRATIONS WERE USED TO CALIBRATE THE ELECTRODES BY SCANNING ACROSS A VOLTAGE RANGE TO DETECT CHANGES IN CURRENT THAT INDICATE NITRATE REDUCTION. B) CURRENT WAS FOUND TO BE LINEARLY RELATED TO CHANGES IN CONCENTRATION. C) AND D) SHOW THE SAME RESULTS OVER A GREATER CONCENTRATION RANGE.....	31
FIGURE 3-14: TESTS TO DETERMINE WHICH FABRICATION METHOD PROVIDES THE BEST SEAL. A) WIRE PUNCHED THROUGH DEVICE. B) WIRE IS CURED IN PDMS MOLD AND THEN PUSHED THROUGH A THIN LAYER OF PDMS THAT DEVELOPS BETWEEN THE WIRE AND THE WALL OF THE CHANNEL. C) WIRE WAS INSERTED INTO THE MOLD WITH 3D PRINTED RUBBER GASKETS.....	32
FIGURE 4-1: INTERIOR OF THE AUTOMATED PLATFORM	34
FIGURE 4-2: PLATFORM COMPONENTS MOUNTED ON THE LID OF THE BOX. SIGNIFICANT COMPONENTS FROM LEFT TO RIGHT: ARDUINO UNO, MOTOR CONTROLLER, TEMPERATURE SENSOR, AND HUMIDITY SENSOR.	34
FIGURE 4-3: ORGANIZATIONAL SCHEMATIC DEPICTING THE PLATFORM’S COMPONENTS AND THEIR PURPOSE.	34
FIGURE 4-4: VOLTAGE DIVIDER CIRCUIT CREATED WITH ELECTRODE EMBEDDED IN THE MINI-CHANNEL DEVICE. THE DOTTED ORANGE LINE SIGNIFIES THE POTENTIAL CONNECTION, THROUGH THE GROWTH MEDIUM, BETWEEN THE PRIMARY ELECTRODE AND GROUND WHEN THE DEVICE IS FULL.	35

FIGURE 4-5: PLATFORM EXPERIMENT — LEFT DEVICE CONTAINS THE CORN ROOT THAT WAS ELECTROCUTED BY THE ELECTRODES.	
THE RIGHT DEVICE WAS USED FOR THE NITRATE MEASUREMENTS DISPLAYED IN FIGURE 4-6.	39
FIGURE 4-6: CORN NITRATE CONSUMPTION	40
FIGURE A-1: SEEDS ARE PLACED ON THE GERMINATION PAPER WHICH IS THEN ROLLED UP AND PLACED IN A FLASK CONTAINING	
CACL ₂ , AND IS USED TO GROW CORN FOR CONTROL TESTS.....	44
FIGURE A-2: DEGASSING CHAMBER.....	44
FIGURE A-3: OXYGEN PLASMA CLEANER	44
FIGURE A-4: PLATFORM PACKED FOR TRANSPORT. FLUID PUMPS ARE MOUNTED ON THE OUTSIDE ALONG WITH A POWER STRIP	
AND THE BREADBOARD WITH THE CONTROL SWITCHES.....	45
FIGURE A-5: PERISTALTIC PUMP FLOW RATE WAS EXPERIMENTALLY DETERMINED, AND THEN NON DIMENSIONALIZED TO FIND	
THE PUMP'S FLOWRATE IS LINEAR W.R.T TIME AS LONG AS THE DUTY CYCLE IS NOT TOO SMALL. THE PLATFORM USES A	
DUTY CYCLE OF 0.5. THE TREND LINE EQUATION ALLOWS THE VOLUME OF PUMPED FLUID TO BE DETERMINED AS LONG AS	
PUMPING TIME IS MEASURED.	46
FIGURE A-6: MATLAB CAMERA CALIBRATION APP DETECTING THE GRID POINTS AND VERIFYING THEIR POSITION TO DETERMINE	
LENS DISTORTION.	47
FIGURE A-7: MATLAB CAMERA CALIBRATION APP FOUND THE POSITION OF THE GRID IMAGES, WHICH ARE USED TO	
DETERMINE AND REMOVE LENS DISTORTION EFFECTS.....	48
FIGURE B-1: MATLAB VALIDATION IMAGE 1	49
FIGURE B-2: MATLAB VALIDATION IMAGE 2	49
FIGURE B-3: MATLAB VALIDATION IMAGE 3	50

LIST OF TABLES

TABLE 2-1: MATLAB VALIDATION: AVERAGE PERCENT ERROR VS. NOMINAL ROOT LENGTH TOTAL DATA SET (EXTREMA REMOVED).....	15
TABLE 4-1: COLOR CODE MEANING RGB LED	38
TABLE B-1: MATLAB VALIDATION: PERCENT ERROR VS. NOMINAL	51

CHAPTER 1. INTRODUCTION

1.1. Literature Review: Plant Root-Environment Interactions

Plants are an important part of our lives; they provide us with shelter, oxygen through photosynthesis, and food. A season's crop yield can be significantly affected by local environmental factors such as disease, pests, microbes, fertilizer, and water availability [1]. Farming practices currently focus on using fertilizer and pesticides, monitoring water availability, crop rotation, and genetic modification of the plant to increase crop yield and stabilize yearly performance [2]. In 2015, the United States produced 13.6 billion bushels of corn and was only 4 percent below expected [3]. However, at the same time there are about 800 million people in the world who do not have enough to eat [4]. With such a large number of people undernourished, an increasing world population and environmental changes, increasing crop yield and plant durability becomes graver.

The effectiveness and sustainability of current farming practices could be greatly improved by properly understanding the interactions between plant roots and their local environment, or rhizosphere [5]. This point is well summarized by Badri *et al.*, "The root system, which was traditionally thought to provide anchorage and uptake of nutrients and water, is a chemical factory that mediates numerous underground interactions" [6]. It then stands to reason that studying these interactions could help drastically boost crop yields while helping crops thrive in previously hostile environments. Crop management will need to include a mixture of traditional farming methods and rhizosphere management to make their crops more durable and to increase crop yields.

Fertilizers are used to supply plants with nutrients lacking in the local environment. When used properly, they can help increase crop yields while making the food itself healthier to eat. The problem arises when overused; they can become toxic and damage the local plant life [7]. Pesticides are extremely useful for warding off pests or harmful microorganisms. When used indiscriminately, they can harm helpful microbes in the soil [8]. These microbes help crops survive through flooding, temperature extremes, high salinity, or even certain types of soil contamination [1].

Microbes working in synergy with hosts is not uncommon after all. Humans have a greater number of microbes living inside them than number of cells in their body [9]. These microbes perform tasks critical to human health, such as assisting in preventing illness and digesting food. Evolving along with humanity, the microbes have been extensively studied in an effort to keep people healthy. Similarly, plants have a collection of beneficial microbes in a region surrounding their roots called the rhizosphere [10]. When humanity began to domesticate crops, and transplant them to new regions, the plant's native microbial relationships were not well considered. Plants naturally developed to flourish in a mixed environment of other local plants, with each plant having a symbiotic relationship with different microbes. Growing only one type of plant per field deprived the plants of these relationships, leaving them weaker and less productive [11, 12].

One of the best examples of plant-microbe symbiosis is how plants rely on bacteria called Rhizobia (*Rhizobiaceae*) to convert atmospheric nitrogen into a form usable by plants [13]. Most plants can only access this nitrogen after the bacterium dies, but plants from the legume family, such as soybeans, can work with the bacteria while it

is alive, greatly improving the process [14]. Systems such as crop rotation or companion planting attempt to take advantage of naturally developed symbiotic relationships like with the legume family and Rhizobia. Crop rotation involves switching growing seasons between different crops to prevent depletion of critical soil resources such as nitrogen or phosphorus. Plants exude chemicals to mediate their interactions with bacteria like Rhizobia in the rhizosphere. Some examples of these interactions are shown in Figure 1-1.

Advancements in rhizosphere management for specific crops, such as corn, could effectively improve crop yields without requiring costly enhancements such as genetic

Figure 1-1: Plant roots exude chemicals to mediate their interactions with microorganism in the rhizosphere [15].

modification, fertilizer, and pesticides. Little is known about the interaction between plant roots and their associated microorganisms. Studying plant's rhizospheric interactions is challenging because many soil microorganisms cannot be easily cultured in laboratories [16]. This is compounded by the fact that the roots are underground, making it difficult to effectively manipulate the local soil environment or observe any interactions. Additionally, different plants also have different reactions with the same microorganism. For rhizosphere management to become a more effective method of improving crop yields on a commercial scale, specific plants need to be studied for their root's unique interactions with different rhizospheric conditions. For this to happen, a practical method of studying how plant roots interact with their environment needs to be developed.

1.2. Literature Review: Microfluidic Study of Plants

There exists prior research involving the application of microfluidic devices to study roots [9, 17-20]. Polydimethylsiloxane (PDMS) is a curable silicone material widely used in soft lithography for micro and nanoscale fluidic devices and is biocompatible [18]. The PDMS channels were created by curing PDMS on a mold, a process called soft lithography. The molds are created using photolithography, a process used to produce features with nanometer precision. The research involves purpose made, micrometer scale, devices to study root phenotype and root interactions with environmental modifiers. *Arabidopsis thaliana* (diameter $\cong 100\mu\text{m}$) is commonly used as a model plant because of its quick life cycle, its ability to produce many offspring, and its status as the first plant to have its genome fully sequenced [21, 22]. Some of the

aforementioned devices will be discussed below to clarify this project's design influences.

1.2.1. RootChip

The RootChip was one of the earliest works to use a microfluidic device to image cellular reactions of multiple *A. thaliana* roots in parallel, to allow for large sample sizes. The plants are grown in microfluidic with channels for root growth. Each root's environment can be controlled separately using miniature valves. The valves, shown in Figure 1-2, are activated with a small pressure increase along the red lines, which opens or closes corresponding blue chemical input channels.

Figure 1-2: RootChip enables parallel study of plant cellular development with environmental control for individual plant roots [9]. **a)** Plants growing in parallel with control and supply microchannels. **b)** RootChip schematic shows root growing down the curved plastic tip into the main channel for observation.

RootChip was used to study the root cellular physiology of *A. thaliana* by measuring root growth and intracellular sugar levels. Humidity was controlled by sealing the device in with an open reservoir of water. The reservoir's air-exposed surface area is much larger than the device's to ensure it provides the majority of the water necessary to saturate the air, reducing evaporation out of the device. The seed was placed in a seed

funnel that comes out of the body at an angle. The root grew from the stem into the main channel where its growth was observed with an inverted microscope.

1.2.2. Plant Chip

The goal of this device was to develop a microfluidic system to efficiently and accurately phenotype a large number of *A. thaliana* samples [19, 23]. Phenotyping involves studying how plants with known genotypes are compared to the actual observable traits of the plant, such as height or color. Plant phenotyping previously involved soil-grown plants in climate-controlled environments, which is expensive and provides poor clarity below the soil. Plant Chip's design focuses on high throughput phenotyping of the root. This was accomplished using a parallel design, transparent construction, and environmental control.

Soft lithography was used to produce a PDMS mold with micrometer channels. This was then attached to glass using an oxygen plasma treatment. *A. thaliana* has tiny seeds, so manually handling them is time consuming and can damage them. To avoid the issue, Plant Chip uses flow through the channel to automatically load the device. Plant Chip's channel geometry was designed to ensure flow would pull only one seed into each funnel. Figure 1-3(b, c) show flow simulation of how seed trapping occurs as well as

Figure 1-3: Plant Chip [19]. **a)** Device schematic showing parallel design. **b)** Flow simulation showing automatic seed trapping. **c)** Experimental verification of seed trapping.

experimental verification of the process. The device holds the seedlings vertically to prevent abnormal root growth due to gravity. The system's low impact on *Arabidopsis* growth and optical clarity allowed for clear phenotyping of multiple plants at once.

1.2.3. Chemical Stimulation of Discrete Root Sections

This microfluidic device was developed to only apply a chemical concentration to a small region, 10 to 800 micrometers, of an *A. thaliana* root. The device was made using PDMS and agar, a common biological culture media. Two layers of PDMS were sandwiched together to create the device. The device was used to test *A. thaliana*'s reaction to growth stimulants and inhibitors.

Figure 1-4: Chemical stimulation over small root segments [18]. **a)** Laminar fluid flow is used small regions of the root. **b)** Florescent microspheres demonstrate flow across a targeted region of the root. (Scale bar is 200 μm)

1.3. Experimental Objectives and Goals

Improving crop yields will help provide more food to the undernourished worldwide. The goal of this research project was to develop an automated mini-channel platform to simplify the study of plant roots with their environment. The literature review

yielded crucial information about the status of fluidics-based root research using microfluidic devices. Corn was chosen as a model plant because of its broader significance as a food source, particularly in the Midwest.

An automated platform was developed to easily characterize root's interactions with their environment. The whole system is sealed inside a box with a large open reservoir to maintain constant humidity. The platform takes images of the seedlings every two hours to track their growth. The plants were grown in a mini-channel device fabricated out of PDMS. The bottom of the device was bonded to a glass slide to ensure the root is clearly visible. The program automatically determines the root's average diameter and contour length. Electrodes are used to check the growth medium levels and ensure the sample stays properly hydrated. The amount of added growth medium is measured. The automated platform also measures temperature and humidity to verify that the environment remains consistent during tests. Another set of specialized electrodes are used to measure the nitrate concentration near the roots. The developed platform will provide researchers with a time efficient method to monitor a plant's interaction with its environment, both microbial and chemical.

CHAPTER 2. METHOD: ROOT GROWTH

CHARACTERIZATION

The corn root's growth is characterized by measuring their contour length and diameter frequently. This is accomplished by taking a picture of the root and using a custom MATLAB program to process the image, returning the root's diameter and contour length. Corn was grown on germination paper as a control group, and was used as a reference point for comparison.

2.1. MATLAB Image Analysis

Roots do not grow straight. Taking reliable length and diameter measurements of a curved object is complicated. Manual measurement of a curved object can only find the straight distance between any two points. It would require multiple measurements of each root as it curves, to accurately determine its length this way, which is time consuming. To streamline the process and improve accuracy, a MATLAB program was developed to automatically measure the root's contour length and diameter. Contour length is the length of the root if it was fully extended to its maximum length. Since the thickness of some plant roots can be inconsistent, average diameter will be used as a metric for comparison. In order to ensure measurements taken from the pictures were reliable, every image had a scale and the camera was held perpendicular to the sample surface. A commented copy of all the MATLAB code is included in Appendix C.

The first step in processing the image is to crop it to only include the sample and the image scale (Figure 2-1a). Once the image is cropped, a millimeter/pixel scale is

Figure 2-1: MATLAB image processing steps.

developed and the RGB image is converted to a black and white binary image. The black and white binary image should only contain the corn root. The binary image is then processed to remove noise that could be misinterpreted as part of the root by the MATLAB program. Small gaps or objects below a certain connectivity threshold are removed. The irregularities on the roots edge are removed using morphological image processing that removes irregularities smaller than a circle with a radius of three pixels.

The program now has a simple binary image that contains a single object that is essentially the 2D shadow of the root (Figure 2-1b). The code determined boundary of the root is traced on top of the original image to enable visual verification root boundary tracking (Figure 2-1c). At this point image is processed two different ways, using either the row progression method or the Euclidean distance transform method. Both methods begin by determining the outer boundary of the root, but they differ in determining the root's centerline (Figure 2-1d).

2.1.1. Row Progression: Approximate Method

The row progression method is an approximate method for finding the root's contour length and diameter. It examines each row in the binary image separately and works its way down the image. The binary image is a matrix composed entirely of ones and zeros, with ones representing only the root. The boundary of the root is determined by removing all zero elements (black background) out of each row, leaving only the one values (white root shadow). The middle one value is the midpoint pixel. The position difference between the farthest left and farthest right one pixel is the diameter value for the row. The midpoint pixel location, and diameter value are recorded, and the process is repeated. All of the collected midpoint pixels represent the root's centerline (Figure 2-1c). The contour length is calculated by adding up the distance from one midpoint pixel in a row to the next, using the Pythagorean distance formula.

The boundary roots are displayed for the user to verify the program is properly determining boundaries. The row progression method is efficient and reliable; however, it doesn't take the curvature of the root into account. Figure 2-2 demonstrates that only the left and right boundaries of the root are considered. This is not much of a problem for

Row progression (RP) method

Figure 2-2: Row progression method cannot accurately process samples with high curvature.

Rows one through four, but Row five would read an incorrect value. The row progression method's difficulties with curvature led to the development of the Euclidean distance transform method.

2.1.2. Euclidean Distance Transform: Exact Method

The row progression method's dependency on moving row by row means that it cannot easily consider the impact of the root's curvature on its measurements. This was accomplished by finding the Euclidean distance transform (EDT) of the binary root image. The EDT finds the distance between every pixel and the nearest nonzero pixel in the binary image. In order for this to work the binary image must be inverted so that the root is represented by zero values, and the background is a one value. The EDT returns an image with zeros for the background and a distance to the closest edge value for each root

pixel. In order to determine the diameter from this information, we had to find the root's centerline.

Morphological image processing (MIP) is used to determine the root's centerline. MIP uses small structuring elements to efficiently search images to recognize patterns within an image. The structuring element's shape and size are defined, and then it is compared to pieces of the image. This allows images to be easily dilated or eroded. The erosion method was used through MATLAB's "thin" morphological operation to reduce the root (Figure 2-1b) down to its centerline (Figure 2-1c). The thin function removes pixels so that an object without holes shrinks down to an object composed of minimally connected strokes." This results in a binary image that only contains the root's centerline.

The "thin" function does not always successfully reduce the root down to its centerline; odd root geometries cannot be reduced down to a single line and branching line occur, causing the code to fail. The contour length can be determined from the centerline image by finding the image's area, due to it being a continuous line only one pixel thick. The root's radius is determined by using element-wise multiplication to map the EDT matrix onto the centerline. This gives each centerline pixel a value representing the closest root edge or radius.

2.1.3. Validation

Both the row progression method and the Euclidean method's results are compared against each other during experimental tests. Since both methods sometimes fall short for different reasons their results will be compared during experimental tests to easily identify image processing errors. This involves measuring pieces of wire while they are straightened to determine their contour length. The wire is then deformed and

evaluated using the MATLAB image analysis program as shown in Figure 2-3. Figures B-1, B-2, and B-3 in Appendix B show the validation images used to test the code's ability to track curved objects.

Root Boundary Trace for Visual Verification

Figure 2-3: Deformed wire used to validate MATLAB code performance.

The average percent error for both image processing methods is shown in Table 2-1. Sample with percent error greater than one hundred are shown on the table as extrema removed. Both the row progression method and the Euclidean method show about fifteen percent error for diameter. The larger diameter error is likely from poor contrast between sample and background and shadows across the sample. These lead to the program choosing an incorrect threshold for converting to binary. While lighting issues also affect the length measurement, the root's longer length reduces their effect. This will be corrected by using a backlight to illuminate the roots from behind, improving

contrast between the root and the background. Another source of error comes from the camera not being perfectly normal to the sample's plane. This causes the pixel to millimeter ratio to vary throughout the image. There is also some human error present, as a human operator selects a scale line for the operator. The approximate method has a larger percent error than the exact method, this is because the approximate does not consider curvature when calculating contour length. The length values are similar enough that they can be compared to each other during experiments as an active form of validation against image processing errors. Samples with abnormal root geometries can be easily identified if the two methods show large divergences. The full validation table is included in Appendix B, Table B-1.

Table 2-1: MATLAB validation: average percent error vs. nominal root parameters, Total data set (Extrema removed)

	% Error Diameter	% Error Length
Euclidean Method	15.97 (15.20)	3.60 (3.50)
Row Progression Method	13.46 (15.23)	71.21 (9.94)

2.2. Control Group

A control test was developed to determine the mini-channel device's impact on corn's growth rate. The control tests use germination paper to grow corn in an easily replicable, standard environment. The corn seeds are kept in a dark envelope in a refrigerator. The dark, cool, and dry environment keeps the seeds in good condition and prevented them from germinating in storage.

2.2.1. Test Procedure

In order to start a control test the seeds were first removed and soaked in a 5% by volume bleach solution to remove the fungicide off their surface. The seeds were then

soaked in water and wiped off to remove and remaining fungicide and rinse away the bleach solution. While the seeds were soaking, germination paper is soaked in tap water. The seeds are placed about 1.5 inches from the top and 1 inch apart on the germination paper (Figure 2-4), which is then rolled up into a cylinder. The cylinder of germination paper is then inserted into a flask containing 3 to 5 inches of the growth medium (Appendix A, Figure A-1) The germination paper will draw up the growth medium and ensure the paper stays wet enough for the plants to grow without drowning them.

The growth medium used is 5 mM of CaCl_2 plus 10 mM NO_3 for some of the tests. Nitrogen is a common plant fertilizer, and it was included to determine corn's nitrate consumption with nitrate sensing electrodes. The entire assemblage is wrapped

Figure 2-4: Corn was grown on germination paper for the control test.

with a plastic bag and placed in a dark environment; this is done to control humidity and promote root growth. A picture is taken of the seedlings daily for six days because, after

six days the corn begins to outgrow the germination paper. Root length and diameter are determined with the developed image processing MATLAB program.

2.2.2. Control Test Results

Figure 2-5 shows the growth of corn root grown in germination paper. The root's growth was recorded daily using a camera. Measurements were taken from the pictures measured using the MATLAB program discussed in the next section. The corn plants beginning to show noticeable germination after day two. After day two, the root's length about doubled every day. The corn grown with NO_3 added to the medium grew faster. The root's diameter increased very slightly over the six days. The small diameter change is comparable with the image processing program's percent error, making it outside the capabilities of the current image processing code. As a result, only root contour length will be quantitatively discussed further.

Figure 2-5: Control test root growth. CaCl_2 and $\text{CaCl}_2+\text{NO}_3$ growth mediums are compared.

CHAPTER 3. METHOD: MINI-CHANNEL DEVICE

In order for the platform to function properly, a mini-fluidic device to grow the root in was developed. This chapter discusses the project's progression through multiple versions of mini-channel devices. It begins with a discussion of necessary design constraints and how these affected design decisions. The discussion then moves toward how the devices were developed, built, and improved to enable root-environment characterization.

3.1. Version 1: Prototype Device

3.1.1. Mini-Channel Device Goals and Design

The mini-channel device's primary goal was to grow corn from a seed and observe how it responded to its environment. This meant it had to keep the corn alive in a visible and controllable environment. The corn seed is housed in a receptacle on top of the body of the device. The receptacle is inserted into the top of the PDMS device. The root grows down the receptacle into the body of the device and into the main root channel. There are two inlets, one for growth medium and another for controlling the root's environment. There is also an outlet to remove fluid from the device. A prototype design, shown in Figure 3-1, demonstrates these properties.

Figure 3-1: Rendering of prototype mini-channel device. PDMS bonded to glass, which is shown black for improved visibility.

The literature review proved the value of PDMS, and it was used for the primary body of the device. The PDMS (Sylgard 184 Silicone Elastomer Kit, Dow Corning) molds, for most of the reviewed plant phenotyping microfluidic devices, were made using soft lithography, an expensive and slow process. Since a goal of the system is to provide a cost efficient method of observing roots, a different method of mold production was chosen. Corn roots have a much larger diameter than *A. thaliana*, 1 mm vs. 100 μm . The roots also grow much faster; a one-week old *Arabidopsis* plants are around 20 mm long, whereas a one-week old corn seedlings are 150 mm long (Figure 2-5). To accommodate the root, the main channel is 3.5 mm wide and 60 mm long end to end. These dimensions are reflected in the PDMS mold (Figure 3-2).

Figure 3-2: Dimensioned top down view of the PDMS mold.

The larger size of corn roots made the microscale resolution of photolithography unnecessary. The mold for the PDMS device was instead made using a 3D printer and designed with SolidWorks. 3D printing can create a greater variety of objects, has a faster turnaround, is cheaper, and requires none of the dangerous chemicals involved with lithography. The PDMS body was then bonded to a glass microscope slide (75 mm x 25 mm), using oxygen plasma bonding to seal the channel. An isometric view of the PDMS mold developed device is shown in Figure 3-3.

Figure 3-3: Isometric view of mini-channel PDMS mold.

The devices were placed in a dark, sealed box to prevent changes in humidity and light from affecting growth. One of our collaborators, Dr. Aaron Lorenz, provided corn seed samples (Hybrid corn seed, Syngenta) and advised 0.5 mM CaCl_2 as a growth medium.

3.1.2. Fabrication and Design Development

The PDMS mold was printed using a 3D printer (Objet 500 Connex3, Stratasys) with a proprietary rigid opaque plastic from Stratasys' Vero family of in-house plastics. The support material was removed using a small water pressure washer. The mold is shown in Figure 3-4.

The literature review discovered that PDMS does not easily cure on certain 3D printed plastics. This was corrected by first sonicating the mold in a 7% by mass dilution of NaOH to remove any remaining support material. The mold was then coated with a perfluorinated trichlorosilane (T2492-KG, UCT Specialties, LLC) which is commonly used in soft lithography to prevent adhesion to the mold. The chemical was applied by placing the mold along with 200 μL of T2492-KG on top of approximately 800 μL

Figure 3-4: Prototype 3D printed mold

mineral oil in a desiccation chamber, or vacuum chamber (Figure A-2). The vacuum pump was run for two minutes and then shut off, with the device left inside for another two hours. This allowed the anti-adhesion chemical to aspirate and coat the mold. The mold can now properly cure PDMS.

PDMS was prepared by mixing ten parts by weight of the silicone elastomer base with 1 part by weight of the elastomer curing agent. The vacuum chamber was used for removing air bubbles, which can reduce clarity and create pockets of air. Such air pockets could interfere with bonding the PDMS body to a glass slide, causing leaks. After degassing the PDMS was poured into the 3D printed mold. The mold was then placed in an oven at 60°C for 4 hours. Once fully cured, the PDMS body was removed from the mold, and a 0.75 mm biopsy punch was used to create inlet and outlet holes.

The PDMS body was then bonded to a glass microscope slide (75 mm x 25 mm), using oxygen plasma bonding to seal the channel. The oxygen plasma bonding was accomplished by placing both a glass slide and the PDMS body into the chamber in the plasma cleaner (PDC-32G, Harrick Plasma, Figure A-3). A vacuum pump evacuated the chamber for four minutes and then the plasma cleaner was turned on for one minute while allowing a small amount of air to re-enter the chamber. After this process was complete, the glass and PDMS body were pressed together. This process causes covalent bonding between the PDMS body and glass slide, basically gluing them together.

The seed funnel was fabricated out of a five mL disposable transfer pipet (VWR). Its top was removed, and it was trimmed to be 10mm shorter. This pipette was used as the seed funnel and inserted into a hole left by the mold at the top of the PDMS body.

Small pieces of stainless steel tubing were inserted into the inlets and outlet and attached to plastic tubing for growth medium and chemical control with a syringe.

3.1.3. Experimental Results

The goal of these devices is to test whether or not corn can be grown in them. Multiple tests were run; the seed or seedling was first placed in the seed funnel pointed down toward the main channel. The devices were then filled with the CaCl_2 growth medium until the top of the seed was covered. The devices were then placed in a dark container to be checked daily. Each day the growth of the root was recorded, and the CaCl_2 was replaced. A white LED backlight (Medium 23 mm x 75 mm, Adafruit) was placed below the device for pictures to provide a strong contrast between the background and root.

The devices proved adept at growing corn. Figure 3-5 demonstrates a device supporting growth for five days. The corn plant started growing on the third day and outgrew the channel by day 5. The root was clearly visible through the glass slide. However, the PDMS became dirty and grew opaque as the experiment progressed. The root funnel made the devices difficult to handle and leaked a considerable amount of fluid near its base. The devices also leaked a smaller amount from the metal tubing inlet and outlets. This meant the devices needed to be refilled at twice a day, which was time-consuming. The varying fluid level could have been affecting the root's development and would interfere attempts at environmental control. The PDMS mold was only capable of being used a few times before the heat caused it to deform, rendering it useless.

Figure 3-5: Prototype device growing corn root over 5 days.

3.2. Version 2: Dual Glass Plate

3.2.1. Design

This version of the mini-channel device was designed to be easier to image, be leak-less, and be more durable. The device mold was thickened to increase its resistance to deforming in the oven. The seed funnel was removed, because it caused too much leaking and couldn't be firmly secured to the relatively soft PDMS. In order to prevent leaking the device's design was simplified. Instead of placing the seed in an external housing, it was placed inside the device's PDMS body. This was accomplished by sandwiching it between two glass plates.

This also made images of the root clearer while making the device sturdier. The primary source of leaking was eliminated by removing the seed funnel. The seed was instead placed inside the PDMS body which was bonded between two glass slides, as shown in Figure 3-6.

Figure 3-6: Two glass plates sandwich the device's PDMS body between them, providing improved clarity and durability.

The devices will need to be held vertical to insert the corn seeds from the top. This design change had the added benefit of allowing gravity to affect the root in the proper direction. The main channel now widens at the top to form a funnel, supporting the seed. Leaking from the metal tubing connectors was reduced by inserting the plastic tubing directly into the device. The larger outer diameter of the tubing produces a better seal. The outer walls of the mold have had their thickness increased to reduce deformation of the mold with multiple uses.

Figure 3-7: Dual plate 3D printed PDMS mold

Figure 3-8: Completed dual plate device.

3.2.2. Fabrication and Design Development

The fabrication steps for the dual plate device are the same as before in section 3.1. However, a glass slide is bonded to both sides of the device.

3.2.3. Experimental Results

Initial attempts at germinating the seed inside the device were met with extremely low germination rates. It was determined that the corn seeds were drowning due to a lack of oxygen. This is because few plants or seeds remain viable underwater for very long, and corn drowns easily compared to other plants [24]. Test procedures were changed to allow seeds to germinate for two days outside the device. This allowed them to begin germinating and growing a root before they were inserted in the device (Figure 3-9).

Figure 3-9: Dual plate device growing corn over the course of 4 days.

Germinating the corn outside the device had the added benefit of allowing seedlings with abnormally fast or no growth to be screen out. Corn's quick growth rate means it often outgrows the device by the fourth day. With the new testing procedures, the corn grown demonstrated growth rates consistent with the control tests (Figure 3-10). Corn's growth rate was drastically affected by the addition of NO_3 to the growth medium. This sensitivity was investigated using specialized nitrate sensing electrodes. Determining corn's exact nitrate consumption facilitate more efficient use of fertilizers. The PDMS molds still deformed due the heat of the oven and clamp pressure. With the device successfully germinating corn, the next version focused on adding the electrodes for measuring fluid level and nitrate concentration.

Figure 3-10: Corn grown the Dual glass plate devices has a growth rates consistent with the control experiments.

3.3. Version 3: Electrodes

3.3.1. Design

This version of the device was designed to include nitrate sensing electrodes, developed by our collaborators in the Lai Lab at UNL. The electrodes can measure the nitrate concentration of the growth medium to determine how much nitrogen the root consumes. Including electrodes in the device required silver and platinum wire to penetrate the PDMS body and stick out into the main channel.

The Objet 3D printer is capable of printing with multiple different materials, one of which is rubber. To prevent any PDMS from curing around the tip of the wire, small rubber sockets were embedded in the mold (Figure 3-11). The embedded electrodes had to exit the side of the mold, where there was a wall. This issue was corrected by rotating the orientation of the mold to have the electrodes point upward. To seal in the PDMS a lid with 3D printed gaskets was designed to be clamped to the mold body. The new top of the device features an additional piece used to hold the electrodes in position during curing.

Figure 3-11: Version 3 Electrode PDMS mold. The mold body (green) gives the PDMS its shape, and has three small sockets to receive embedded electrodes. The is a lid (blue), with an embedded gasket. The red piece holds the electrodes in position while the PDMS cures. While curing in the oven the red piece faces upward, to prevent PDMS from spilling out around the wires.

3.3.2. Fabrication and Design Development

Fabrication steps for this version of the device involve the same preparation steps as versions one and two. Before pouring PDMS into the mold the wires are inserted into the sockets. The mold's lid is then clamped on. The PDMS can then be poured. The final step is to place the wire holder on top of the device. This mold was used to create devices for the automated platform.

The 3D printed mold still showed deformations over time, and to correct this the outside body of the mold was made out of aluminum (Figure 3-12a, b). The main channel in the mold is still made using the 3D printer, because the rubber wire sockets are very useful and this piece showed no signs of thermal deformation. The channel is removable, and held in place by fitting into a small groove in the aluminum. It is prevented from falling out of the groove when the lid is clamped on. The lid and body of the mold have groves

that fit together preventing PDMS from leaking out. The aluminum mold proved very durable and easy to use. The aluminum mold is simpler to use. Its major components do not need to be coated to cure PDMS making it safer to handle. Additionally, the center channel comes out of the mold with the PDMS, which reduces the risk of damaging the device during removal.

Figure 3-12: Aluminum mold a) The mold showing PDMS with embedded electrodes, before removal from the mold. Demonstrating how the mold is used. b) A cross section view of the mold shows how the plastic channel fits into a groove in the metal mold. It also shows how the gaskets widen to ensure they are not pulled out of the mold.

3.3.3. Experimental Results

Three electrodes are used to measure nitrate (NO_3) concentrations in the mini-channel device, two silver and one platinum [25]. The electrodes were developed by the Lai Lab at UNL. One of the silver wires is the working electrode, and is treated to form an oxide that improves nitrate reduction on its surface. The second silver wire is called the reference electrode, and is used with the working electrode to measure the potential created by nitrate reduction. The reference electrode is used to measure potential, but no current passes through it. The platinum wire is called the counter electrode. Current created by nitrate reduction is measured as it passes through the working electrode and

the counter electrode. Figure 3-13 shows how the current generated by nitrate reduction responds to the applied voltage potential. Calibrating the electrodes determined they have a linear current response to changes in nitrate concentration.

Electrode Calibration

Figure 3-13: Calibrating the nitrate concentration measuring electrodes. **a)** Solutions containing different known nitrate concentrations were used to calibrate the electrodes by scanning across a voltage range to detect changes in current that indicate nitrate reduction. **b)** Current was found to be linearly related to changes in concentration. **c)** and **d)** show the same results over a greater concentration range. (Figure courtesy of the Lai Lab at UNL)

Figure 3-14: Tests to determine which fabrication method provides the best seal. **a)** Wire Punched through Device. **b)** Wire is cured in PDMS mold and then pushed through a thin layer of PDMS that develops between the wire and the wall of the channel. **c)** Wire was inserted into the mold with 3D printed rubber gaskets.

The nitrate sensing electrodes required a tight seal around the wire to give accurate results., which proved difficult. Figure 3-14a shows that punching the electrode wires through the side of the PDMS tore a piece of PDMS out of the main channel wall. This caused the device to leak, compromising the sample. To attempt to correct this, the wires were pushed up against the PDMS mold wall and cured inside the PDMS. After curing, a small layer of PDMS still remained. Pushing the wire through caused a small amount of tear out. While not enough to cause leaking, it would have affected the electrode's performance (Figure 3-14b). The rubber sockets proved very effective, because the electrodes had a good seal and didn't have any PDMS on them (Figure 3-14c).

CHAPTER 4. METHOD: PLATFORM DEVELOPMENT

4.1. Platform Design and Function

Manually imaging plant roots over multiple days is not efficient for high sample volumes. To make studying root-environment interactions more efficient, an automated platform was developed. It is capable of controlling the plant's environment and measuring its response to environmental changes. The platform is housed inside a sealed dark box to insulate the roots from changing ambient conditions. This also makes the platform easy to transport. The automated system was operated using MATLAB.

The platform can grow corn in two mini-channel devices at once, to increase throughput (Figure 4-1). The two devices are managed by the MATLAB program separately from one another. MATLAB's USB webcam toolbox is used to easily control a webcam to image samples. The Arduino toolbox uploads a script onto Arduino board (Figure 4-2) turning it into a live remote client for MATLAB to control. This simplified the platform's development as MATLAB automatically handled communication with the Arduino. An organizational schematic of the platform is shown in Figure 4-3. The nitrate electrodes require a potentiostat with micro-amp resolution, which is not controlled by the system. The other components are controlled through the Arduino Uno, a programmable microcontroller capable of interacting with digital and analog sensors.

Figure 4-1: Interior of the automated platform

Figure 4-2: Platform components mounted on the lid of the box. Significant components from left to right: Arduino Uno, motor controller, temperature sensor, and humidity sensor.

Figure 4-3: Organizational schematic depicting the platform's components and their purpose.

4.1.1. Growth Medium Management

In order to measure the growth medium, two additional electrodes are embedded in the device's PDMS body. Growth medium is pumped into the device using a peristaltic pump (Figure A-4) controlled by a motor controller. The motor controller is necessary as the pumps require more power than the Arduino on its own can supply.

The voltage divider circuit, shown in Figure 4-4, has the growth medium pulled to ground. As a result, when the device is full, a portion of the current from the source

Figure 4-4: Voltage Divider Circuit created with electrode embedded in the mini-channel device. The dotted orange line signifies the potential connection, through the growth medium, between the primary electrode and ground when the device is full.

grounded causing the Arduino to read a low, zero in binary. If fluid does not cover the primary electrode, no current is directed through the fluid to ground and the Arduino reads the full five volts and returns a high, or one value. If the electrodes read a high value, then the driver program knows the device is low and it turns on the peristaltic pump to refill the device. The pump continues until the fluid reaches the electrode, completing the circuit and the code stops the pump. The pumping time is recorded and the volume of pumped fluid is calculated. The equation used to find total pumped volume was experimentally found by measuring the time it took the pump to move 100 mL (Figure A-5, Duty Cycle=0.5).

The automated platform waits two hours between tests, because the electrodes develop an oxide if used too frequently. The oxide increases their resistance causing it to permanently read as a high value, this causes the peristaltic pumps to turn on and overflow the mini-channel device. The backup electrode is added to protect against overflow, and is wired directly onto the primary electrode. This allows it to function without any additional programming. The growth medium level is checked before any other functions as the program loops, to ensure measurements are taken when the device is full.

4.1.2. Root Imaging

The automated platform uses a USB webcam (HD-5000, Microsoft), and MATLAB's USB webcam support package, to take images of the root. The webcam package allows for control over a USB webcam, which are cheap and provide sufficient image quality.

Because all camera lenses have innate lens distortions that can affect measurements. Camera Calibrator, a MATLAB tool, is used to remove these distortions. A grid pattern, known by the calibration program, is moved in front of the camera. (Figure A-6 & A-7). The program knows the grid's pattern and its size is provided by the user, thus it can determine the location and extent of the lens distortions. Once the distortions are known, the program outputs script that can be included in other MATLAB functions to remove the distortion's effect. This allows for more accurate measurements.

Before a picture is taken, the Arduino is used to turn on an LED backlight to illuminate the root in the dark box. This increases the contrast between the root and background, which helps the image analysis program properly determine the root's edge. After pausing a moment to adjust, the webcam takes an image of both mini-channel devices and saves the image. This image is sent through the image analysis subroutine, which uses the row progression method and Euclidean method to determine the root's contour length. The binary, traced boundaries, and centerline images for both methods and devices are saved.

4.1.3. Ambient Environment Measurements

Temperature (TMP102, Sparkfun) and humidity (HIH-4030, Sparkfun) sensors were used to verify that conditions in the box remain relatively constant. Both sensors are controlled through the Arduino by MATLAB. Temperature is measured first, because relative humidity is dependent on temperature. The humidity sensor outputs an intermediate sensor relative humidity value. The true relative humidity is found using a correction equation provided by the company and the measured temperature.

4.1.4. User Control, Device Feedback and Data Management

The platform provides user control through two switches mounted on the top of the box, one to pause the program and another to end it. The program will continue running as long as the power switch remains on. The state of both the pause loop and end loop are checked after the program saves the current loop's results. If the pause switch is active, then the program will remain in a small pause loop until it is turned off. All images and data are exported to a csv file at the end of each loop to prevent data loss if there is an error. If the power switch is active, then the program will quit. A RGB (Red, Green, Blue) LED is used to provide feedback on what the platform is currently doing. Its color patterns and their meaning are shown in Table 4-1.

Table 4-1: Color Code Meaning of RGB LED

LED Color – Duration/Pattern	Platform Action
Blue - Steady	Operating Normally
Green – Short flash	Webcam imaging root
RGB – Quickly flashing between colors	Pumping fluid into device 1 or 2
RGB – Slowly flashing between colors	Pause loop active
Red – Steady	Power loop active, program ended

4.2. Experimental Results

The individual components in the automated platform were all tested independently. To test the whole platform, corn roots were grown inside the mini-channel device and automatically observed over two days. A solution of 5 mM CaCl_2 and 10 mM NO_3 was used. The corn root appeared to be attracted to electrical current and grew

towards the electrodes. The electrodes used to measure fluid level had to be removed. The corn root grew towards the electrodes and caused the circuit to short through the corn seedling, killing it. The corn root killed by the current is shown in Figure 4-5.

Figure 4-5: Platform Experiment – Left device contains the corn root that was electrocuted by the electrodes. The right device was used for the nitrate measurements displayed in Figure 4-6.

Two successful measurements were achieved without the automated fluid level monitoring of the system. Only two successful measurements were taken due to time constraints and the plant and evaporation caused the electrodes to become uncovered (Figure 4-5). The corn root also grew onto the electrodes, disrupting their sensitivity. Figure 4-6 shows a reduction in the peak current a few hours after placing the seed. The reduction in current correlates to a reduction in nitrate concentration. Imperfections in the oxide coating the wire mean exact concentration could not be determined, but the measurements can be qualitatively compared. Some growth medium had already evaporated out of the device, and since only the water evaporates, not the nitrate, an

increase in peak current and concentration could be expected. However, due to the consumption of nitrate by the seed we can see a decrease in the peak current over time.

Figure 4-6: Nitrate current measurement decreased over five hours and fifteen minutes, displays corn plant's consumption of nitrate.

4.3. Future Platform Improvements

The platform's current layout is small and crowded, this makes changing any of the components difficult. The platform has a lot of wires that are organized by taping them down. While convenient, the tape has to be removed to make any changes. More adhesive wire holders of different sizes should be used. The number of wires traveling throughout the system could be reduced by using a more efficient means of communicating to the sensors.

I²C uses a data wire and clock wire to stagger communications between different sensors. This allows them to all use the same two wires to communicate to the Arduino.

The platform's box size is convenient for transportation, but there is very little working room inside, and a larger box would be recommended.

The platform will need a higher resolution camera to get a clear picture of any interactions. The camera will need a close focal range and high resolution. It should be compatible with MATLAB's USB camera toolbox, as it makes integrating the camera into the system simple. A fish eye lens could be attached to a high resolution webcam and the image undistorted using MATLAB's camera calibration software. The fish eye lens would allow the camera to be positioned very close to the mini-channel device.

In order to measure the device's growth medium level, the electrodes will have to be replaced with an alternative method of fluid level measurement. The webcam could be used determine the fluid level, or pressure gauge could be embedded in the device. The nitrate sensing electrodes will also have to be protected from the corn plant's roots growing onto them and damaging them. This could be done by having the electrodes positioned behind a permeable membrane.

CHAPTER 5. CONCLUSION

This project developed an automated platform with the purpose of streamlining research exploring plant root's interactions with their environment. The platform can automatically measure root growth while simultaneously controlling environment. The only user input required is to commence or terminate experiments. The platform's affordable construction and adaptable design means it can be adapted to study a large variety of root-rhizosphere interactions. As understanding of culturing microbes improves, the image processing capabilities can be expanded to include microbial tracking.

Corn's selection as a model plant allowed the mini-channel device's size to be increased relative to conventional microfluidic devices. The device molds could then be produced with a 3D printer, lowering fabrication costs. Using a 3D printer instead of photolithography allowed for a broader range of PDMS mold designs. This design freedom was utilized to embed rubber gaskets in the PDMS mold.

Corn was grown in the mini-channel device without a large deviation from normal growth. The corn seed was found to consume an observable amount of nitrate over time. Root contour length was accurately determined, but certain geometries caused image processing errors. An increase in measurement accuracy and reliability can be obtained by finding a more robust method of determining the root's centerline.

By automating root-rhizosphere interactions, while lowering costs and improving design freedom, the developed platform simplifies observing root-rhizosphere

interactions. This will enable other researchers to focus on observing new plant-rhizosphere interactions without developing a custom fluidic device first.

APPENDIX.A SUPPLEMENTAL FIGURES

Figure A-1: Seeds are placed on the germination paper which is then rolled up and placed in a flask containing CaCl_2 , and is used to grow corn for control tests

Figure A-2: Degassing chamber

Figure A-3: Oxygen Plasma Cleaner

Figure A-4: Platform packed for transport. Fluid pumps are mounted on the outside along with a power strip and the breadboard with the control switches.

Figure A-5: Peristaltic pump flow rate was experimentally determined, and then non dimensionalized to find the pump's flowrate is linear w.r.t time as long as the duty cycle is not too small. The platform uses a duty cycle of 0.5. The trend line equation allows the volume of pumped fluid to be determined as long as pumping time is measured.

Figure A-6: MATLAB Camera Calibration App detecting the grid points and verifying their position to determine lens distortion.

Figure A-7: MATLAB Camera Calibration App found the position of the grid images, which are used to determine and remove lens distortion effects.

APPENDIX.B MATLAB IMAGE ANALYSIS VALIDATION

Figure B-1: MATLAB Validation Image 1

Figure B-2: MATLAB Validation Image 2

Figure B-3: MATLAB Validation Image 3

Table B-1: MATLAB Validation: Percent Error vs. Nominal

MATLAB Validation: Percent Error v.s. Nominal									
Nominal		Exact Method				Approximate Method - No Curvature			
Length	Diameter	Average Minimum Diameter	Diameter % Error	Contour Length	Length % Error	Average Diameter	Diameter % Error	Contour Length	Length % Error
70.00	2.75	2.30	16.42	71.04	1.49	2.13	22.64	75.52	7.89
101.00	2.75	2.16	21.59	102.48	1.46	2.95	7.17	180.86	79.07
69.00	2.75	2.12	23.07	70.50	2.18	2.90	5.62	194.68	182.15
160.00	3.55	2.76	22.39	166.16	3.85	3.09	12.85	176.15	10.09
82.00	3.55	2.94	17.30	84.34	2.85	3.18	10.41	213.23	160.03
70.00	2.75	2.22	19.17	71.72	2.46	1.96	28.57	75.68	8.11
150.00	2.75	2.15	21.74	154.05	2.70	2.18	20.72	165.25	10.17
101.00	2.75	2.05	25.34	103.94	2.91	2.12	23.04	173.84	72.11
69.00	2.75	2.05	25.31	70.33	1.93	2.13	22.70	181.30	162.75
160.00	3.55	2.48	30.25	166.45	4.03	2.61	26.57	179.43	12.14
82.00	3.55	2.79	21.48	86.44	5.42	2.82	20.48	177.58	116.56
70.00	2.75	2.86	4.02	73.16	4.52	2.68	2.61	76.77	9.68
150.00	2.75	2.89	5.09	155.31	3.54	2.87	4.36	164.38	9.58
101.00	2.75	2.92	6.18	105.31	4.27	2.94	6.77	173.52	71.81
69.00	2.75	2.95	7.18	72.24	4.69	2.97	8.12	179.17	159.66
160.00	3.55	3.64	2.50	168.65	5.41	3.67	3.49	178.93	11.83
82.00	3.55	3.63	2.39	88.16	7.52	3.65	2.71	186.14	127.00
Average:			15.97		3.60		13.46		71.21
Average: Extrema Removed			15.20		3.50		15.23		9.94
Note: 1 extreme data sat with an % error greater than 100									

APPENDIX.C MATLAB CODE

C.A MATLAB AUTOMATION MAIN DRIVER PROGRAM

```

%% MATLAB Driver program

% This is the primary control program for the automated system. It
% controls all MATLAB functions as well as an Arduino Uno.

%% Operational Notes
%{
-USB Webcams and MATLAB support package for Arduino must be installed
for
program to function.

%}
%% Program Initialization
clc
clear
close all

% Set loop count to zero
count=1;

% Set pause count to zero
pcount=0;

% Choose configuration based on different computer
desktop=1; %1 is desktop, =0 is laptop
if desktop==1
a = arduino('COM3','UNO'); %Desktop uses COM port 3
else
a = arduino('COM4','UNO'); %Laptop uses COM port 4
end
% Initialize arduino
% Initialize exit loop condition x=0. x=1 when switch is flipped and
the
% program will exit. y is the same except it is for pausing the program
x=0;
y=0;
%% Arduino Uno pin definitions

% 3-way switch
switch_pin='D2';
configurePin(a, switch_pin, 'DigitalInput')

% 3 way switch 2
switch2_pin='D7';
configurePin(a, switch2_pin, 'DigitalInput')

% White LED Backlights
% Backlight 1

```



```

backlight_1_pin='D8';
configurePin(a, backlight_1_pin, 'DigitalOutput')

% Backlight 2
backlight_2_pin='D4';
configurePin(a, backlight_2_pin, 'DigitalOutput')

% RGB LED (Digital IO pins used not PWM for RGB LED)
red_pin='D10';
configurePin(a, red_pin, 'PWM')
green_pin='D9';
configurePin(a, green_pin, 'PWM')
blue_pin='D11';
configurePin(a, blue_pin, 'PWM')

% Fluid pumps
% Pump 1
pump_1_pwm_pin='D5';
configurePin(a, pump_1_pwm_pin, 'PWM')
pump_1_d_pin='D6';
configurePin(a, pump_1_d_pin, 'DigitalOutput')

% Pump 2
pump_2_pwm_pin='D3';
configurePin(a, pump_2_pwm_pin, 'PWM')
pump_2_d_pin='D12';
configurePin(a, pump_2_d_pin, 'DigitalOutput')

% Water electrodes
% Currently just designated for A1, A2, A3
electrode_1='A1';
configurePin(a,electrode_1,'DigitalInput')

electrode_2='A2';
configurePin(a,electrode_2,'DigitalInput')

electrode_3='A3';
configurePin(a,electrode_3,'DigitalInput')

% Humidity Sensor
humidity_pin='A0';
configurePin(a,humidity_pin,'AnalogInput')

%% Configure temperature sensor TMP102 with I2C
% http://www.mathworks.com/help/supportpkg/arduinoio/examples/measure-temperature-from-i2c-device-on-arduino-hardware.html?refresh=true
% Scan for available I2C addresses
% ddrs = scanI2CBus(a)='0x48'

%Create the I2C temperature device object
tmp102 = i2cdev(a, '0x48');

% i2cdev with properties: Pins: A4(SDA), A5(SCL); Bus: 0; Address: 72 (0x48)
%% Configure webcam objects

```

```

% http://www.mathworks.com/help/supportpkg/usbwebcams/ug/acquire-
images-from-webcams.html
% Use webcamlist to determine webcam numbers prior to experiment.
Internal
% laptop webcams will be on the list. Look for 'Microsoft® LifeCam HD-
3000'

if desktop==1
    cam1_num=2; %desktop
else
    cam1_num=3; %laptop
end

% Create a webcam object

% Microsoft® LifeCam HD-6000 for Notebooks

cam1=webcam('Microsoft');
cam1.Resolution='1280x720';
cam1.BacklightCompensation=1; %Disable = 0, Enable = 1

cam1.Brightness=90; %Range: 30-255
cam1.Contrast=5; %Range: 0-10
cam1.Exposure=-9; %Range: -11-1
cam1.ExposureMode='auto'; % Enable automatic exposure mode
cam1.Focus=17; %Range: 0-40; Set focus, as the distance to the
optimally focused target, in millimeters.
cam1.FocusMode='manual';
cam1.Pan=0;
cam1.Saturation=83; %Range: 0-200
cam1.Sharpness=50; %Range: 0-50
cam1.Tilt=0;
cam1.WhiteBalance=4500;
cam1.WhiteBalanceMode='auto'; % Enables automatic white balancing
(color temperature)
cam1.Zoom=0;

%% Camera Setup for each experiment
% img=snapshot(cam1);
% pause(1)
writeDigitalPin(a, backlight_1_pin, 1);
writeDigitalPin(a, backlight_2_pin, 1);
pause(1)
cam1.ExposureMode='auto';
pause(5)
img=snapshot(cam1);
[experiment_name,mm_per_pixel,mm_scale,
line_scale,ylim1,xlim1,ylim2,xlim2,image_device]=camera_setup(cam1);
writeDigitalPin(a, backlight_1_pin, 0);
writeDigitalPin(a, backlight_2_pin, 0);

%% Webcam Calibration - 5.5 mm grid used

```

```

% http://www.mathworks.com/help/vision/ug/single-camera-calibrator-
app.html
% Auto-generated by cameraCalibrator app on 19-Mar-2016
%-----
if desktop==1 %Desktop webcam calibration

% Auto-generated by cameraCalibrator app on 30-Mar-2016
%-----

% Define images to process
imageFileNames = {'C:\Users\bioflow
lab\Documents\Automation_Kevin_Kreis\Webcam Calibration\notebook webcam
calibration HD 6000\Image1.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image2.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image3.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image5.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image9.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image10.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image11.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image12.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image13.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image14.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image15.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image18.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image20.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image23.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image24.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image27.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image28.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image29.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image30.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image31.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image32.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image33.png',...

```

```

'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image34.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image35.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image37.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image38.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image39.png',...
'C:\Users\bioflow lab\Documents\Automation_Kevin_Kreis\Webcam
Calibration\notebook webcam calibration HD 6000\Image40.png',...
};

% Detect checkerboards in images
[imagePoints, boardSize, imagesUsed] =
detectCheckerboardPoints(imageFileNames);
imageFileNames = imageFileNames(imagesUsed);

% Generate world coordinates of the corners of the squares
squareSize = 5.500000e+00; % in units of 'mm'
worldPoints = generateCheckerboardPoints(boardSize, squareSize);

% Calibrate the camera
[cameraParams, imagesUsed, estimationErrors] =
estimateCameraParameters(imagePoints, worldPoints, ...
    'EstimateSkew', false, 'EstimateTangentialDistortion', false, ...
    'NumRadialDistortionCoefficients', 2, 'WorldUnits', 'mm', ...
    'InitialIntrinsicMatrix', [], 'InitialRadialDistortion', []);

% View reprojection errors
h1=figure; showReprojectionErrors(cameraParams, 'BarGraph');

% Visualize pattern locations
h2=figure; showExtrinsics(cameraParams, 'CameraCentric');

% Display parameter estimation errors
displayErrors(estimationErrors, cameraParams);

% For example, you can use the calibration data to remove effects of
lens distortion.
originalImage = imread(imageFileNames{1});
undistortedImage = undistortImage(originalImage, cameraParams);

% See additional examples of how to use the calibration data. At the
prompt type:
% showdemo('MeasuringPlanarObjectsExample')
% showdemo('StructureFromMotionExample')

else %laptop webcam calibration
% Auto-generated by cameraCalibrator app on 23-Mar-2016
%-----

% Define images to process

```

[illegible]

```

};

% Detect checkerboards in images
[imagePoints, boardSize, imagesUsed] =
detectCheckerboardPoints(imageFileNames);
imageFileNames = imageFileNames(imagesUsed);

% Generate world coordinates of the corners of the squares
squareSize = 5.500000e+00; % in units of 'mm'
worldPoints = generateCheckerboardPoints(boardSize, squareSize);

% Calibrate the camera
[cameraParams, imagesUsed, estimationErrors] =
estimateCameraParameters(imagePoints, worldPoints, ...
    'EstimateSkew', false, 'EstimateTangentialDistortion', false, ...
    'NumRadialDistortionCoefficients', 2, 'WorldUnits', 'mm');

% View reprojection errors
h1=figure; showReprojectionErrors(cameraParams, 'BarGraph');

% Visualize pattern locations
h2=figure; showExtrinsics(cameraParams, 'CameraCentric');

% Display parameter estimation errors
displayErrors(estimationErrors, cameraParams);

% For example, you can use the calibration data to remove effects of
lens distortion.
originalImage = imread(imageFileNames{1});
undistortedImage = undistortImage(originalImage, cameraParams);

% See additional examples of how to use the calibration data. At the
prompt type:
% showdemo('MeasuringPlanarObjectsExample')
% showdemo('SparseReconstructionExample')

end
%% Storage Variable Setup

%j=number of saved variables
j=19;
results1=zeros(j,1);
results2=zeros(j,1);

%% Choose file save location

start_path='C:\Users\Kevin\OneDrive\Graduate
Research\Automation\Programming\MATLAB';
folder_name=uigetdir(start_path, 'Choose save location for results. ');
test_gap=input('How many minutes between tests? ');

% End of program setup

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%

%% Continuous Loop with Exit Switch
%%As long as switch is set set to 0 then the loop will continue. Once
the
switch is flipped at set to 1, then the program will stop.
while x == 0
    %% Pause Loop
    switch2_state=readDigitalPin(a,switch2_pin);
    %%Check time
    format shortg
    c=clock;
    year=c(1);
    month=c(2);
    day=c(3);
    hour=c(4);
    minute=c(5);
    seconds=c(6);
    %% Pause condition
    if switch2_state == 0

        % Set LED to blue to signify the program is running
        led_color(a,red_pin,green_pin,blue_pin,0,0,1)
        % Read electrode's status 0=water (wet), 1=no water (dry)
        electrode_1_state=readDigitalPin(a,electrode_1);
        electrode_3_state=readDigitalPin(a,electrode_2);
        electrode_2_state=readDigitalPin(a,electrode_3);

        %% Manage water level
        % Refill device 1
        p1_c1=clock;
        while electrode_1_state~=0
            led_color(a,red_pin,green_pin,blue_pin,1,1,1)
            % Set pump direction CCW=0, CW=1
            writeDigitalPin(a,pump_1_d_pin,0)
            % Turn on pump
            writePWMDutyCycle(a,pump_1_pwm_pin,0.5)
            % pause(.5)
            % Check current fluid levels
            electrode_1_state=readDigitalPin(a,electrode_1);
        end
        led_color(a,red_pin,green_pin,blue_pin,0,0,0)
        % Turn off pump
        writePWMDutyCycle(a,pump_1_pwm_pin,0)
        led_color(a,red_pin,green_pin,blue_pin,0,0,1)

        % Calculate volume pumped - pump 1
        format shortg
        p1_c2=clock;
        time_diff_pump_1(count)=etime(p1_c2,p1_c1);
        volume_1(count)=0.2199*time_diff_pump_1(count)+1.3706;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
format shortg
p2_c1=clock;

% Refill device 2
while electrode_2_state~=0
    led_color(a,red_pin,green_pin,blue_pin,0,1,1)
    % Set pump direction CCW=0, CW=1
    writeDigitalPin(a,pump_2_d_pin,0)
    % Turn on pump
    writePWMDutyCycle(a,pump_2_pwm_pin,0.5)
    % pause(.5)
    % Check current fluid levels
    electrode_2_state=readDigitalPin(a,electrode_3);
end
led_color(a,red_pin,green_pin,blue_pin,0,0,0)
% Turn off pump
writePWMDutyCycle(a,pump_2_pwm_pin,0)
led_color(a,red_pin,green_pin,blue_pin,0,0,1)

% Calculate volume pumped - pump 2
format shortg
p2_c2=clock;
time_diff_pump_2(count)=etime(p2_c2,p2_c1);
volume_2(count)=0.2199*time_diff_pump_2(count)+1.3706;
%% Image Aquisition
% Turn on white LED backlight x 2
writeDigitalPin(a, backlight_1_pin, 1)
writeDigitalPin(a, backlight_2_pin, 1)
cam1.ExposureMode='auto';

% Pause to allow webcams to adapt to change in light
pause(5)

% Take pictue w/ MATLAB controlled webcam
img=snapshot(cam1);
pause(0.5)
led_color(a,red_pin,green_pin,blue_pin,0,1,0)
pause(0.5)
led_color(a,red_pin,green_pin,blue_pin,0,0,0)
%Undistorts image
img = undistortImage(img, cameraParams);

% Save webcam image
camera='original';
image_name='img';
baseFileName =
sprintf('%d_%s_%s_%s__day%d__hour%d__minute%d.png',count,experiment_n
ame,image_name,camera,day,hour,minute);
fullFileName = fullfile(folder_name, baseFileName);
imwrite(img, fullFileName);

% Turn off white LED backlight x 2

```



```

writeDigitalPin(a, backlight_1_pin, 0)
writeDigitalPin(a, backlight_2_pin, 0)

%% Process Image
ylim=ylim1;
xlim=xlim1;
camera='device1';
figure(1)

results1(1:8,count)=image_analysis(cameraParams,experiment_name,camera,
c,folder_name,count,day,month,hour,minute,seconds,caml,mm_scale,line_scale,
img,mm_per_pixel,ylim,xlim,image_device);
ylim=ylim2;
xlim=xlim2;
camera='device2';
figure(2)

results2(1:8,count)=image_analysis(cameraParams,experiment_name,camera,
c,folder_name,count,day,month,hour,minute,seconds,caml,mm_scale,line_scale,
img,mm_per_pixel,ylim,xlim,image_device);

%% Read Temperature Sensor in Celsius
write(tmp102, 0, 'uint8');
data = read(tmp102, 2, 'uint8');
temperature = (double(bitshift(int16(data(1)), 4)) +
double(bitshift(int16(data(2)), -4))) * 0.0625;

%% Read Humidity Sensor
% Sensor datasheet:
https://www.sparkfun.com/datasheets/Sensors/Weather/SEN-09569-HIH-4030-datasheet.pdf
vSupply=5; %supply voltage
vRef=5; %reference voltage
humidity_voltage=readVoltage(a,humidity_pin);
Vout=humidity_voltage;

%https://github.com/angryelectron/tweetpot/blob/master/arduino/HIH4030/
HIH4030.cpp

%Relative Humidity is calculated using the following equations
taken from the datasheet:
% (1) Vout = (VSupply) (0.0062(sensorRH) + 0.16)
% (2) trueRH=(sensorRH)/(1.0546-0.00216*temperature)
%temperature
% is in degrees celcius

% Solving (1) for sensor RH which can be plugged into (2)
% sensorRH=161.29*Vo/Vs-25.81

% Convert voltage measurment into relative humidity
sensorRH=161.0*Vout/vSupply-25.8;
% Compensate reading for proper temperature
trueRH=sensorRH/(1.0546-0.00216*temperature);

```

```

% Image processing returns first 8 results

%% Write Results

% Results for device 1
results1(9,count)=temperature;
results1(10,count)=sensorRH;
results1(11,count)=trueRH;
results1(12,count)=pcount;
results1(13,count)=count;
results1(14,count)=month;
results1(15,count)=day;
results1(16,count)=hour;
results1(17,count)=minute;
results1(18,count)=seconds;
results1(19,count)=volume_1(count);
results1_t(count,:)=results1(:,count).';

% Save results for device 1
data_name='results1_t';
baseFileName = sprintf('%s%s.txt',experiment_name,data_name);
fullFileName = fullfile(folder_name, baseFileName);
dlmwrite(fullFileName,results1_t(count,:), '-append')
%xlswrite(fullFileName,results1)
%       data_name='results1';
%       baseFileName =
sprintf('%d%s%s.xlsx',count,experiment_name,data_name);
%       fullFileName = fullfile(folder_name, baseFileName);
%       xlswrite('test.xlsx',results1)
%cell_1
display_loop_results1=results1(:,count).';

% Results for device 2
results2(9,count)=temperature;
results2(10,count)=sensorRH;
results2(11,count)=trueRH;
results2(12,count)=pcount;
results2(13,count)=count;
results2(14,count)=month;
results2(15,count)=day;
results2(16,count)=hour;
results2(17,count)=minute;
results2(18,count)=seconds;
results2(19,count)=volume_2(count);
results2_t(count,:)=results2(:,count).';

% Save results for device 2
data_name='results2_t';
baseFileName = sprintf('%s%s.txt',experiment_name,data_name);
fullFileName = fullfile(folder_name, baseFileName);
dlmwrite(fullFileName,results2_t(count,:), '-append')

% End of current count loop

```

```

count=count+1;

else
    % Pause loop, when triggered the program remains in this loop
    % flashing the RGB LED.
    pcount=pcount+1;

    led_color(a,red_pin,green_pin,blue_pin,0,1,0)
    pause(1)
    led_color(a,red_pin,green_pin,blue_pin,0,0,0)
    led_color(a,red_pin,green_pin,blue_pin,1,0,0)
    pause(1)
    led_color(a,red_pin,green_pin,blue_pin,0,0,0)
    led_color(a,red_pin,green_pin,blue_pin,0,0,1)
    pause(1)
    led_color(a,red_pin,green_pin,blue_pin,0,0,0)

    %Recheck switch 2 state to determine if code pause should end
    switch2_state=readDigitalPin(a,switch2_pin);
end

% Check exit program condition (On switch state)
switch_state=readDigitalPin(a,switch_pin);
if switch_state==1
    x=1 % Set x=1 to exit the program
    % Red LED for 5 secondss to show the program has exited.
    led_color(a,red_pin,green_pin,blue_pin,1,0,0)
    %pause(5)
    %led_color(a,red_pin,green_pin,blue_pin,0,0,0)
end
pause(60*test_gap)

%Check time
format shortg
c=clock;
year=c(1);
month=c(2);
day=c(3);
hour=c(4);
minute=c(5);
seconds=c(6);

end

% Code to turn off red light
%led_color(a,red_pin,green_pin,blue_pin,0,0,0)

%% Acknowledgments
% Thanks to Troy Anderson and MATLAB Central user "Image Analyst" for
their assistance in creating this program.
% Thanks to Eric Diamond for assistance with this program.

```

C.B LED COLOR FUNCTION

```
%Easily controls LED color for the Experiment Driver Program v2
function [] = led_color(a,red_pin,green_pin,blue_pin,red,green,blue)

% Simplifies LED Control
writePWMDutyCycle(a, blue_pin, 0)
writePWMDutyCycle(a, green_pin, 0)
writePWMDutyCycle(a, red_pin, 0)
if blue==1
    writePWMDutyCycle(a, blue_pin, blue)
end
if red==1
    writePWMDutyCycle(a, red_pin, red)
end
if green==1
    writePWMDutyCycle(a, green_pin, green)
end
end
```

C.C IMAGE ANALYSIS FUNCTION

```

function [img_results] =
image_analysis(cameraParams,experiment_name,camera,c,folder_name,count,
day,month,hour,minute,seconds,caml,mm_scale,line_scale,
img,mm_per_pixel,ylim,xlim,image_device)
% Root Image Analysis 4 converted into a function

%% Process Input Image
img=img;

%Display input image with scaled colors
imagesc(img)

%Convert the image to grayscale
img_gray=rgb2gray(img);

%Modify the image size to only include the root of focus
i=img_gray(ylim,xlim);
i_ref=img(ylim,xlim,:);

%Convert the image to black and white using graythresh
i2=im2bw(i,graythresh(i));

%Remove small objects/artifacts from the image to prevent morphological
%image processing errors
i3=bwareaopen(i2,20);

%Fill in small holes/artifacts in the image to prevent morphological
image
%processing errors
i4=imfill(i3,'holes');

%Morphologically open the image
i5=imopen(i4,strel('disk',3));

% Inverse the image depending on image type (determined by user)
if image_device==1
    i5=1-i5;
end

%% Calculations Section

%Find pixel length of line scale which outputs a matrix of the
following
%form [X1 Y1; X2 Y2]
dy=line_scale(2,2)-line_scale(1,2);
dx=line_scale(2,1)-line_scale(1,1);
scale_length=sqrt(dx^2+dy^2);

%Determine the pixel to mm conversion ratio
mm_per_pixel=mm_scale/abs(scale_length);

```

```

%Determine image size
[m,n]=size(i5);

%% Euclidean Image Transform
%Apply the Euclidean Image Transform (Distance Transform) to show the
%distance from the centerline to the edge of the root

% Inverse the image depending on image type (determined by user)
if image_device ==0
    i5_EDT=1-i5;
else
    i5_EDT=i5;
end

%Apply the euclidean distance transform to the morphologically modified
%binary image
EDT=bwdist(i5_EDT);

%Use morphological image processing to thin the image down to a
minimally
%connected stroke, which is the root's centerline
i_thin=bwmorph(i5,'thin',Inf);

%Use element wise multiplication to map the distance transform onto the
%centerline, giving the shortest distance between the perimeter of the
root
%and the centerline at each given point on the centerline
%http://www.mathworks.com/help/matlab/ref/times.html
i_f=times(EDT,i_thin);

%Find the average distance (average diameter) and standard deviation
for
%the matrix i_f. Mean2 and std2 are the 2D versions of mean and std.
mean_e=mean2(nonzeros(i_f*mm_per_pixel));
mean_times_2_e=mean2(nonzeros(2*i_f*mm_per_pixel));
s_dev_e=std2(nonzeros(i_f*mm_per_pixel));

%When calculating the contour length it is important to notice that
i_thin
%is a single pixel line whose area is equal to the length of the line.
%MATLAB has no built in length functions, however there is a built in
area
%function.
%Both i_f and i_thin return the same area value
%bw_arebwarea(i_thin);
bw_area=bwarea(i_f);
contour_length_e=bw_area*mm_per_pixel;

% Old Approximate Method - Works down image vertically
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% Determine which pixel represents the midpoint of the root
for j=1:m

```

```

        nonZero=find(i5(j,:));
        midpoint_pixel(j,1) = floor(median(nonZero));
    end
    midpoint_pixel(find(isnan(midpoint_pixel))) = [];

    % Plot the midpoint pixel vector
    %figure(3)
    %plot(midpoint_pixel)

    % Calculate the contour length of the root
    for i=2:length(midpoint_pixel)
        delta_y=mm_per_pixel;
        delta_x=abs(midpoint_pixel(i)-midpoint_pixel(i-1))*mm_per_pixel;
        contour_length_vector(i-1)=(delta_y^2+delta_x^2)^(1/2);
    end
    contour_length_a=sum(contour_length_vector);

    % Find average diameter w/out considering curvature
    for j=1:m
        %Determine which pixels reperesent the root in the jth row
        nonZero2=find(i5(j,:));
        %if statement prevents errors when the end of the root is reached
        if isempty(nonZero2)==0
            %Calculate the diameter of the jth row
            diameter(j)=abs((nonZero2(1,end)-nonZero2(1,1)))*mm_per_pixel;
        else
            end
    end
    %Find the overall average diameter of the root
    average_diameter_a=mean2(diameter);
    s_dev_a=std2(nonzeros(diameter));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%
    %% Results and Comparison Plots
    percent_error_contour_length=abs((contour_length_e-
    contour_length_a)/contour_length_e)*100;

    %Subplot(m,n,p) %m and n are different from above, variables reused
    %m-number of subplot grid rows
    %n-number of subplot grid columns
    %p-grid position for new axes (Defines position of plotted image)
    m=1;
    n=4;
    %Graph important images for visual comparison
    %figure(1)
    subplot(m,n,1)
    imshow(i_ref)
    title('Original Cropped Image')

    subplot(m,n,2)
    imshow(i5)
    title('Processed B&W Image')

    subplot(m,n,3)

```

```

imshow(i_thin)
title('Minimally Connected Stroke - Centerline')

%Show code defined root boundaries/perimeter for user visual
verification
%First plot the reference image
subplot(m,n,4)
imshow(i_ref)
title('Root Boundary Trace for Visual Verification')
%Trace the boundaries in the binary image
perims=bwboundaries(i5);

%Plot the boundary points on top of the reference image
hold on
%figure(1)
for i=1:length(perims)
    boundary = perims{i}
    plot(boundary(:,2),boundary(:,1),'g','LineWidth',2);
end
hold off
%Find the perimeter of the root in the binary image
perimeter = bwperim(i5);
%http://www.mathworks.com/matlabcentral/newsreader/view\_thread/300433
%Find the area of the root
area = regionprops(i5,'area');

%Save Images

image_name='fig_subplot';
baseFileName =
sprintf('%d_%s_%s_%s__day_%d__hour_%d_minute_%d.png',count,experiment_n
ame,image_name,camera,day,hour,minute);
fullFileName = fullfile(folder_name, baseFileName);
saveas(gcf,fullFileName)

image_name='i5';
baseFileName =
sprintf('%d_%s_%s_%s__day_%d__hour_%d_minute_%d.png',count,experiment_n
ame,image_name,camera,day,hour,minute);
fullFileName = fullfile(folder_name, baseFileName);
imwrite(i5, fullFileName);

image_name='i_ref';
baseFileName =
sprintf('%d_%s_%s_%s__day_%d__hour_%d_minute_%d.png',count,experiment_n
ame,image_name,camera,day,hour,minute);
fullFileName = fullfile(folder_name, baseFileName);
imwrite(i_ref, fullFileName);

image_name='i_thin';
baseFileName =
sprintf('%d_%s_%s_%s__day_%d__hour_%d_minute_%d.png',count,experiment_n
ame,image_name,camera,day,hour,minute);
fullFileName = fullfile(folder_name, baseFileName);
imwrite(i_thin, fullFileName);

```



```
% Store results to be sent back to main program
img_results=[mean_e,mean_times_2_e,s_dev_e,contour_length_e,average_dia
meter_a,s_dev_a,contour_length_a,percent_error_contour_length];

%http://www.mathworks.com/matlabcentral/answers/244938-image-
processing-finding-length-and-diameter-of-a-curved-object

end
```

C.D CAMERA SETUP FUNCTION

```

function [experiment_name,mm_per_pixel,mm_scale,
line_scale,ylim1,xlim1,ylim2,xlim2,image_device]=camera_setup(cam1)
experiment_name=input('Experiment Name: ','s');

% Take webcam image to determine sample type
img=snapshot(cam1);

% Define image type
image_device=input('If white root on dark background give a 0, if white
root lit up with an LED background give a 1. ');

%Display image
imshow(img)
%Create a rectangle around the target root
%The box should be drawn at the bottom of the seed
title('Draw a rectangle around each root. Then draw the scale line.
Double click to confirm each step');
crop_rect1=wait(imrect);

%Read the rectangle 1 information
ylim1 = floor(crop_rect1(2)):floor(crop_rect1(2))+
floor(crop_rect1(4));
xlim1 = floor(crop_rect1(1)):floor(crop_rect1(1))+
floor(crop_rect1(3));

crop_rect2=wait(imrect);

%Read the rectangle 2 information
ylim2 = floor(crop_rect2(2)):floor(crop_rect2(2))+
floor(crop_rect2(4));
xlim2 = floor(crop_rect2(1)):floor(crop_rect2(1))+
floor(crop_rect2(3));

%% Determine Image Scale
%Create a line to determine the mm to pixel conversion ratio
line_scale=wait(imline);

%Prompt the user for the length of the scale object
mm_scale=input('How many millimeters is the scale? ');

%Find pixel length of line scale which outputs a matrix of the
following
%form [X1 Y1; X2 Y2]
dy=line_scale(2,2)-line_scale(1,2);
dx=line_scale(2,1)-line_scale(1,1);
scale_length=sqrt(dx^2+dy^2);

%Determine the pixel to mm conversion ratio
mm_per_pixel=mm_scale/abs(scale_length);
end

```

REFERENCES

1. Reid, A. and S.E. Greene, *How Microbes Can Help Feed the World of Microbiology Colloquium*. 2012: p. 1-36.
2. Tilman, D., et al., *Global food demand and the sustainable intensification of agriculture*. Proceedings of the National Academy of Sciences, 2011. 108(50): p. 20260-20264.
3. U.S. Department of Agriculture, *Crop Production 2015 Summary*, U.S.D.o. Agriculture, Editor 2016, National Agricultural Statistics Service.
4. Food and Agriculture Organization of the United Nations. *The State of Food Insecurity in the World 2015*. 2015; [World hunger statistics 2015]. Available from: <http://www.fao.org/hunger/en/>.
5. Coleman-Derr, D. and S.G. Tringe, *Building the crops of tomorrow: Advantages of symbiont-based approaches to improving abiotic stress tolerance*. Frontiers in Microbiology, 2014. 5(JUN).
6. Badri, D.V., et al., *Rhizosphere chemical dialogues: plant-microbe interactions*. Current Opinion in Biotechnology, 2009. 20: p. 642-650.
7. Hartikainen, H., *Biogeochemistry of selenium and its impact on food chain quality and human health*. Journal of Trace Elements in Medicine and Biology, 2005. 18(4): p. 309-318.
8. Gunapala, N. and K.M. Scow, *Dynamics of soil microbial biomass and activity in conventional and organic farming systems*. Soil Biology and Biochemistry, 1998. 30(6): p. 805-816.
9. Grossmann, G., et al., *The RootChip: an integrated microfluidic chip for plant science*. Plant Cell, 2011. 23: p. 4234-4240.
10. Buée, M., et al., *The rhizosphere zoo: An overview of plant-associated communities of microorganisms, including phages, bacteria, archaea, and fungi, and of some of their structuring factors*. Plant and Soil, 2009. 321: p. 189-212.
11. Hartmann, A., et al., *Plant-driven selection of microbes*. Plant and Soil, 2009. 321: p. 235-257.
12. Brooker, R.W., et al., *Improving intercropping: a synthesis of research in agronomy, plant physiology and ecology*. New Phytologist, 2015. 206(1): p. 107-117.
13. L E Barber, J.D.T., S A Russell and H J Evans, *Acetylene reduction (nitrogen fixation) associated with corn inoculated with Spirillum*. Applied and Environmental Microbiology, 1976. 32(1): p. 108-113.
14. Extension Agronomists, R.F., and John Idowu, *Nitrogen Fixation by Legumes*, in *Consumer and Environmental Sciences* 2015, New Mexico State University.
15. Bais, H.P., et al., *The role of root exudates in rhizosphere interactions with plants and other organisms*. Annual review of plant biology, 2006. 57: p. 233-266.
16. Malamy, J.E., *Intrinsic and environmental response pathways that regulate root system architecture*. Plant, Cell & Environment, 2005. 28(1): p. 67-77.
17. Agudelo, C.G., et al., *TipChip: A modular, MEMS-based platform for experimentation and phenotyping of tip-growing cells*. Plant Journal, 2013. 73: p. 1057-1068.

18. Meier, M., E.M. Lucchetta, and R.F. Ismagilov, *Chemical stimulation of the Arabidopsis thaliana root using multi-laminar flow on a microfluidic chip*. Lab on a chip, 2010. 10: p. 2147-2153.
19. Jiang, H., et al., *Plant chip for high-throughput phenotyping of Arabidopsis*. Lab on a chip, 2014. 14: p. 1281-93.
20. Parashar, A. and S. Pandey, *Plant-in-chip: Microfluidic system for studying root growth and pathogenic interactions in Arabidopsis*. Applied Physics Letters, 2011. 98: p. 2009-2012.
21. *Arabidopsis: The Model Plant*, The Multinational Coordinated Arabidopsis thaliana Functional Genomics Project. 2013, National Science Foundation.
22. *First-Ever Complete Plant Genome Sequence Is Announced, International team reveals DNA secrets of Arabidopsis thaliana*, Office of Legislative and Public Affairs. 2000, National Science Foundation.
23. Jiang, H., et al., *A Microfluidic Whole-Plant Phenotyping Device* Department of Electrical and Computer Engineering , Iowa State University , Ames , IA , USA Department of Genetics , Development and Cell Biology , Iowa State University , Ames , IA , USA Design of Device Trap. 2013: p. 1539-1542.
24. Morinaga, T., *Germination of Seeds Under Water*. American Journal of Botany, 1926. 13(2): p. 126-140.
25. Mohsin Ahmad Bhat, P.P.I., Vijay Raman Chaudhari and Santosh Krishna Haram, *Mechanistic aspects of nitrate ion reduction on silver electrode: estimation of O–NO₂– bond dissociation energy using cyclic voltammetry*. Royal Society of Chemistry: New Journal of Chemistry, 2009. 33: p. 207-210.