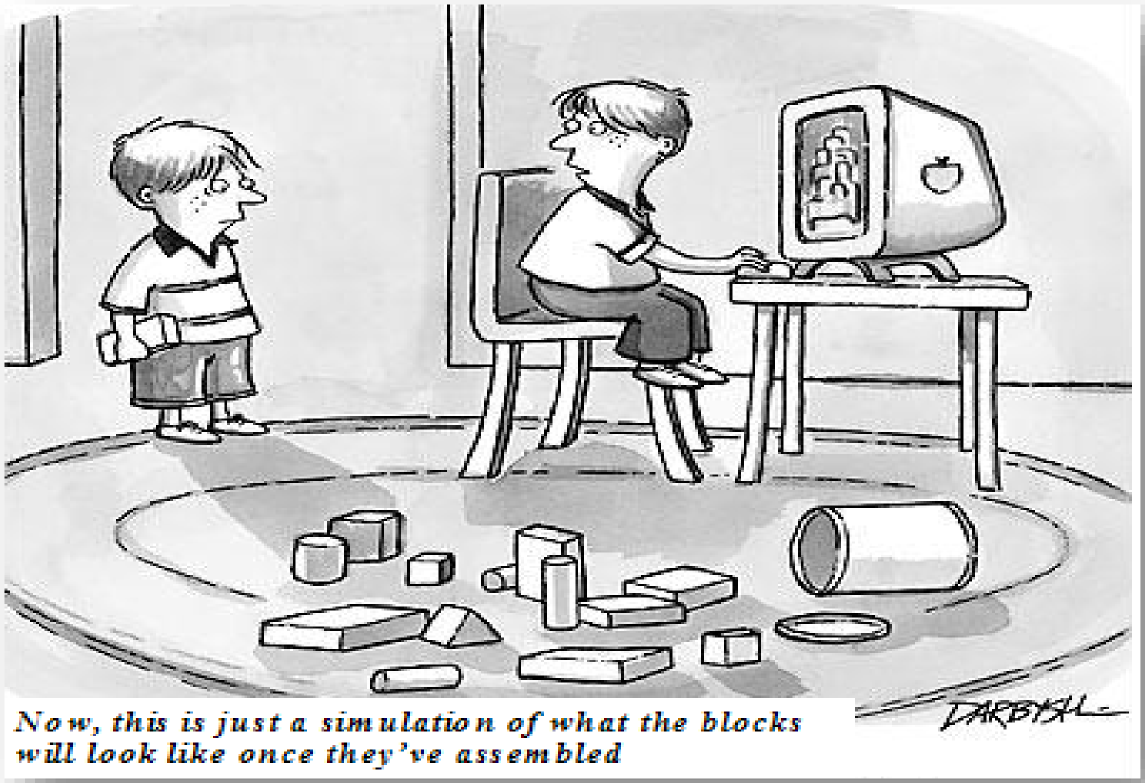
ECE569
Module 10

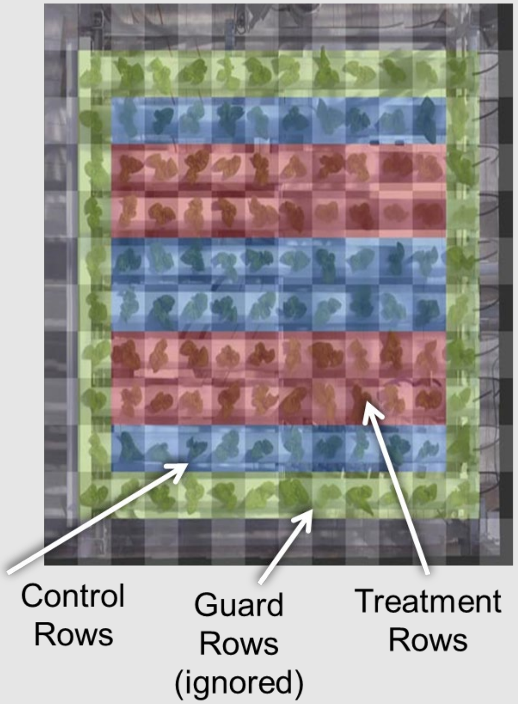

Now, this is just a simulation of what the blocks will look like once they've assembled
DARBYSH

- Debugging, Profiling

Controlled Environment Plant Production

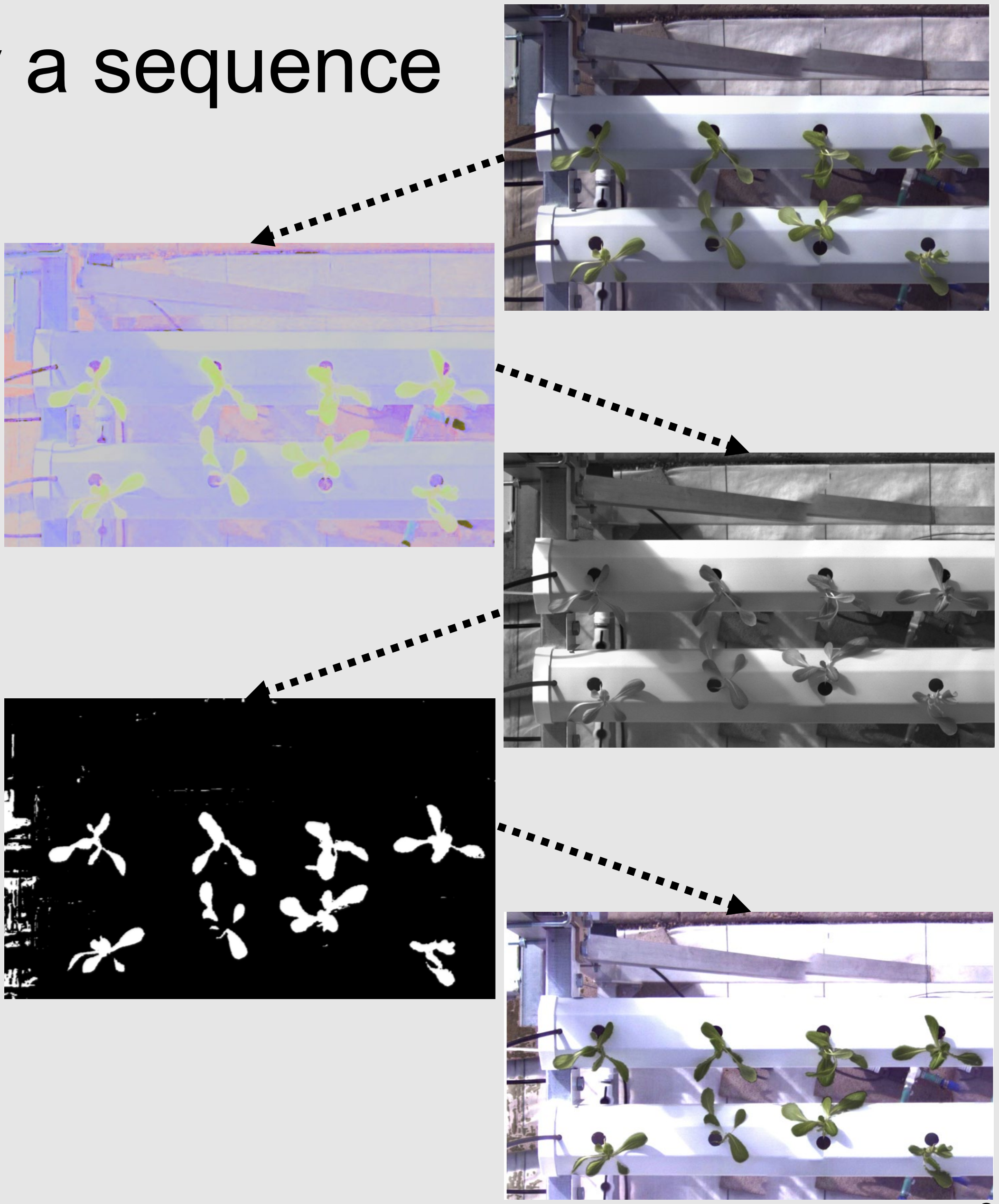
In Collaboration with

UA Controlled Environment Agriculture Center



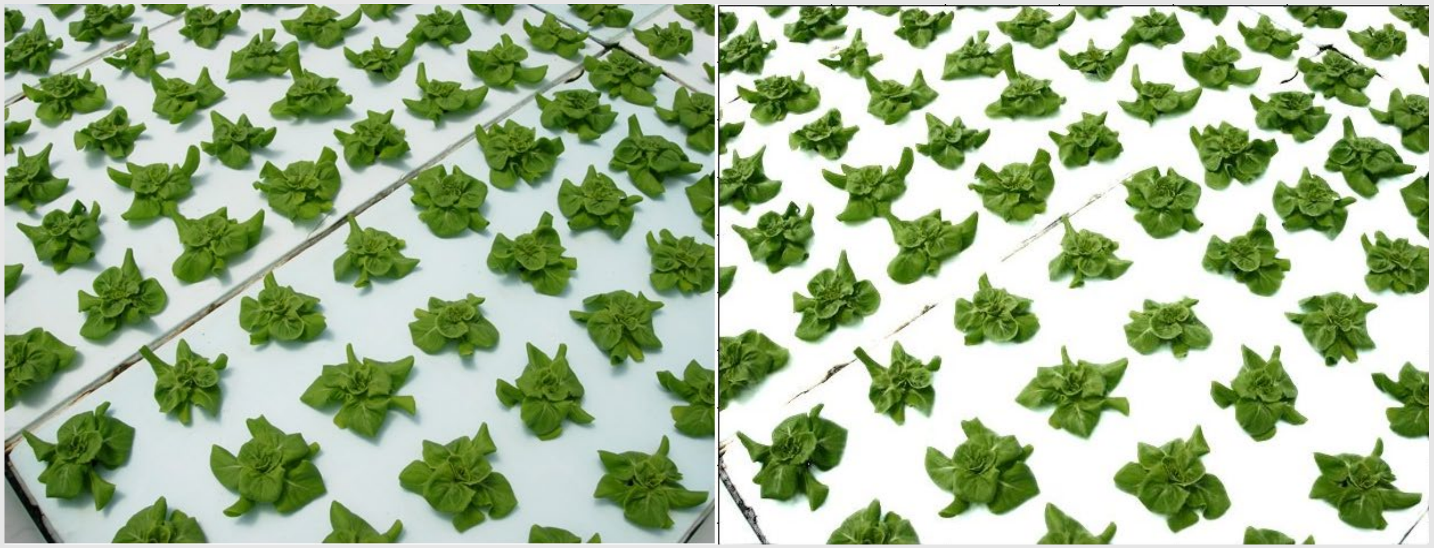
Process

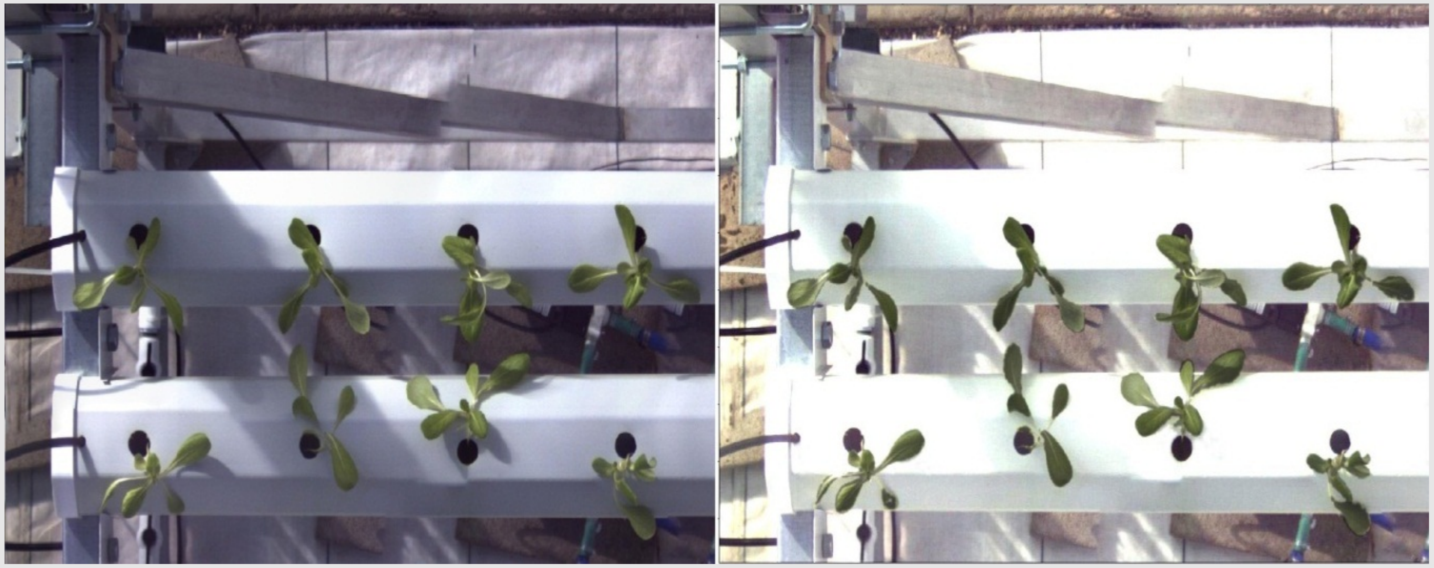
- For each pixel apply a sequence of transformations:
  - Compute color invariant image
  - Compute grayscale image
  - Histogram generation
  - Masking technique with a convolution 5x5 window



Shadow Removal

THE UNIVERSITY OF ARIZONA®





CUDA Toolkit

- Compiler flags
- Debuggers
- Profilers

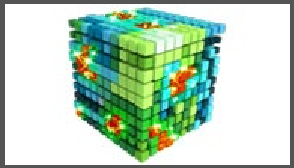
Developer Tools - Debuggers

Nsight

Nsight Systems

CUDA-GDB

CUDA MEMCHECK



NVIDIA Provided

arm FORGE

TotalView®

3rd Party

https://developer.nvidia.com/debugging-solutions

NVCC Compiler

- **NVIDIA provides a CUDA-C compiler**
  - nvcc
- **NVCC compiles device code then forwards code on to the host compiler**
  - Nvcc parses .cu files
  - To compile .c needs to be renamed as .cu
- **Can be used to compile & link host only applications**

Compiler Flags

- **Remember there are two compilers being used**
  - NVCC: Device code
  - Host Compiler: C/C++ code
- **NVCC supports some host compiler flags**
  - If flag is unsupported, use –Xcompiler to forward to host
    - e.g. –Xcompiler –fopenmp
- **Debugging Flags**
  - -g: Include host debugging symbols
  - -G: Include device debugging symbols
  - -lineinfo: Include line information with symbols

CUDA-MEMCHECK

- **Memory debugging tool**
  - No recompilation necessary
  - %> cuda-memcheck ./exe
- **Can detect the following errors**
  - Memory leaks
  - Memory errors (OOB, misaligned access, illegal instruction, etc)
  - Race conditions
  - Illegal Barriers
  - Uninitialized Memory
- **For line numbers use the following compiler flags: -Xcompiler -rdynamic -lineinfo**

http://docs.nvidia.com/cuda/cuda-memcheck

CUDA-MEMCHECK EXERCISE

- **Refer to D2L->Content->Demo->2.Debug->add.cu**

Instructions:

1. Build & Run add.cu using the instructors given in the source code
   Do you get the correct results?
2. Run with cuda-memcheck
   %> cuda-memcheck ./myadd
3. Add nvcc flags “-Xcompiler –rdynamic –lineinfo”
4. Rebuild & Run with cuda-memcheck
5. Fix the illegal write

http://docs.nvidia.com/cuda/cuda-memcheck

CUDA-GDB

- **cuda-gdb is an extension of GDB**
  - Provides seamless debugging of CUDA and CPU code
  - Works on Linux and Mac
- **For a Windows debugger use NVIDIA Nsight Eclipse Edition or Visual Studio Edition**

http://docs.nvidia.com/cuda/cuda-gdb

GDB EXERCISE

- Refer to D2L->Content->Demo->2.Debug->add.cu
- Run cuda-gdb for add.cu and fix the bug.

```
$ cuda-gdb --args ./a.out
```

Run a few cuda-gdb commands:

```
– (cuda-gdb) b main                    //set break point at main
– (cuda-gdb) r                         //run application
– (cuda-gdb) l                         //print line context
– (cuda-gdb) b foo                     //break at kernel foo
– (cuda-gdb) c                         //continue
– (cuda-gdb) cuda thread               //print current thread
– (cuda-gdb) cuda thread 10            //switch to thread 10
– (cuda-gdb) cuda block                //print current block
– (cuda-gdb) cuda block 1              //switch to block 1
– (cuda-gdb) d                         //delete all break points
– (cuda-gdb) set cuda memcheck on      //turn on cuda memcheck
– (cuda-gdb) r                         //run from the beginning
```

Next

- Profilers