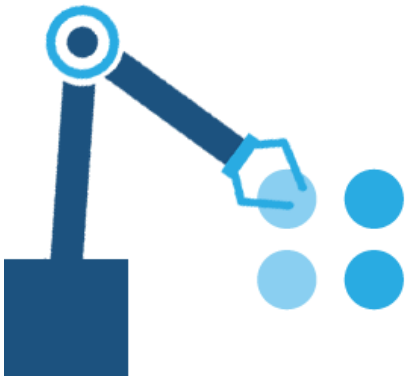
Boston University
EC464 Senior Design Project
Final Prototype Testing Report



NanoPack
A PACKING MACHINE FOR NANOVIEW BIOSCIENCES

BY:

Devin Bidstrup *dbids at bu dot edu*

Joseph Walsh *jwalsh15 at bu dot edu*

Paul Stephen Hutchinson Maltaghati *psmalta at bu dot edu*

Justin Melville *jem2000 bu dot edu*

George Kent-Scheller *georgeks at bu dot edu*

Required Materials:

Hardware:

NANOPACK MACHINE

- LitePlacer Robotics kit
  - All mechanical parts
    - Framing, sliders, connectors, etc.
  - TinyG motor controller
  - USB cameras, LED ring lights
  - Stepper motors (3 NEMA 17, 1 NEMA 14)
  - Belts and pulleys
  - Limit switches
  - Cable chain
  - Vacuum Pump
- 24V power supply
- Wiring & connectors
- E-stop
- Test Computer
- Faux leather green screen

Software:

1. Python Control software
2. Machine learning program to process image from wide angle camera
3. Machine learning program to process image from actuator camera
4. .NET GUI for User Interface

Set Up:

The setup consists of both hardware and software components. From the GUI we will run a control program which will move the machine's actuator from rest to hover over a traveler, whose location will be determined using the large angle camera and a machine learning script. The machine will then use the actuator camera to take a photo and locate chips with the image processing. Then, the actuator will jog down the z-axis, create a seal with the chip using the vacuum pump, and return back to the clearance height. Then, the machine will move to the clamshell location using the object detection machine learning script. From there, the chip will be lowered down into the specified well where the vacuum seal will be broken as the chip makes contact with the sloped surface inside the clamshell slots. Our mechanical goals will be to have the machine moving within the workspace as well as a demonstration of the actuator movement. Our software goals are to locate the traveler and clamshells from the wide angle camera, move to these locations, use number recognition to verify the correct chip number, and lastly pick up and place the chips in their correct location.

Measurable Criteria and Scoring:

The criteria for successful running and output is as follows:

Criteria	Completed (Y/N)
Object recognition and edge detection shows location of chips in traveler	Y
All digits of all chips are successfully extracted	N
All numbers are correctly classified	N
Chips correspond to the information given in the CSV	Y
Software notifies user of chip position error	Y
Machine moves to specific traveler and clamshell locations	Y
Vacuum pump picks up chips	Y
Chips are released in the clamshell slots	Y

Results:

1. Object recognition and edge detection shows location of chips in traveler
   This was done successfully during our demo. The trained YOLO object detection found every single clamshell and traveler with a reasonable level of confidence to be able to correctly classify all of them. After this, the camera moved to the position of the object, and moved towards the x and y edges until it successfully detected the edge it was supposed to find. Once this is doen, we cross-reference the CSV uploaded by the user, and then we know the exact location of the chips.
2. All digits of all chips are successfully extracted
   This is currently having issues due to the chip pictures not being lined up correctly. When creating a test set, the camera was able to line up exactly the same every time, and we were able to achieve 100 percent test accuracy when this occurs. However, when running the program as part of the control loop, the pictures that were taken were significantly less accurate, and this meant that some of the digits were cut off or otherwise not properly visible.
3. All numbers are correctly classified
   Similarly to the previous point, we were able to achieve 100 percent test accuracy yet still ran into issues when running the classification as part of the control loop. This is because of the same reason for the pictures being off-centered. Fixing these two issues will be done in tandem, as fixing one will hopefully fix the other.
4. Chips correspond to the information given in the CSV
   This is done successfully. The GUI takes in a CSV, the format of which is defined by our client, and then feeds this to the control loop. The control loop is then able to compare this information to the number of the chip detected through the camera. If they correspond, then the machine packs the chip and if they do not match, then the user is notified of an error.
5. Software notifies user of chip position error
   This is done successfully. To check this, we loaded a chip into the traveler that does not have a matching location in the csv file. When this mismatch was detected, a message pops up in the GUI warning the user of this issue.
6. Machine moves to specific traveler and clamshell locations
   This is done successfully. One the object detection is finished, we run edge detection on the object. This allows us to find one edge in each of the x and y axes, and thereby allowing us to pinpoint where the corner of the object is. Once this is known, we can move to specific locations within both the traveler and clamshell.
7. Vacuum pump picks up chips
   The vacuum pump works successfully. When we lower it to the proper height and activate the vacuum, the chips are obtained and able to be moved to the desired location in the clamshell.

8. Chips are released in the clamshell slots

We were successful in releasing the chips into the clamshell slot. This process was surprisingly quite simple due to the slant of the slots in the clamshell. When we lower the chip into the correct slot, this slant presses on the chip at an angle, and automatically breaks the seal the vacuum pump has on the chip. This means that we don’t have to worry about calculating the proper drop off height, and can simply lower the chip until the seal is broken automatically.

What's Next

Graphical User Interface

Most of the work for our GUI has been completed. The only things which we have left involve adding features such as alerting the user if no travelers or clamshells are detected, as well as adding additional error detection with regard to potential machine learning issues. Another thing which we want to do is make sure that the GUI is easily portable and that it will work at the NanoView site. We will have to do this once the machine is brought to the client.

Machine Learning Models

At this point, we have now developed machine learning algorithms which detect and classify travelers and clamshells from images as well as chips and chip label numbers. We have also demonstrated that we can move to detected locations. These are some things which we will be doing to make improvements in the final weeks of the project:

a. **Retraining our algorithms with data from new hardware**

   After we experienced some error with our current camera during the final prototype testing, the professors advised that we invest in a more powerful camera. We found cameras online which were meant to be used in image processing, and we think that if we use these cameras, we will have much higher accuracy, especially when it comes to digit detection.

b. **Improving number cropping**

   It appears that most of the issues with the digit detection during the final prototype detection came from the chip number cropping. We miss some of the digits within the crop, and if we improve this crop, we believe that we will have much higher accuracy in corresponding the chip numbers with those in the CSV.

c. **Training the machine learning with the tweezers**

   In the final prototype testing, we decided to use the vacuum pump. The vacuum pump proved to be very efficient at picking up chips. For the final product, though, we still want to use the tweezers since our team designed the actuator. To use the actuator, we will have to move the mechanical constraints and retrain our edge detection.

d. **Doing testing at the NanoView site**

   Once the machine is transported to NanoView, we will have to do some additional testing to ensure that the lighting difference will not be a major issue. It will also be important to explain to the NanoView employees how the machine learning program works in case any error comes up while they are using the product.

TinyG Motor Controller

The TinyG will require additional coordinate tuning to make sure that the tweezers can be utilized. Other than the coordinate tuning, we will have to make sure that all the serial connections work when we transport the machine to NanoView. We had success in connecting to the TinyG during the final prototype testing, and we believe that as long as we can establish successful communication between the computer, the TinyG, and the overhead

camera, we will not experience major issues when we install the machine.

Mechanical

For the mechanical design, the main area of work will be in installing and testing our actuator. While we did have success in picking up chips using the tweezers, the actuator was unable to reach all locations throughout the entire traveler due to the location of the restraints, which is why we used the vacuum pump for the final prototype testing. We now plan on moving the constraints in order to leave enough space for the solenoid. Another issue we will be improving is the final dropping of the chips into the clamshell slots. Sometimes, the tweezers would open with too much force and the chips wouldn’t fully sit in the slots. To address this issue, we plan on moving the point of contact between the tweezers and solenoid.