Automated Mast Sorting Device

Authors: Thomas Howell, Jack Horton, Vincent Wang

Figure 1: Mast sample collected form Duke Forest.

Background

At the base of every forest ecosystem are mast, or the fruits of trees and shrubs, which includes acorns, hickory nuts, and beech nuts (Figure 1). Mast play several fundamental roles in ecology, from sustaining animal populations to influencing reproduction patterns. Mast are everywhere in the Duke Forest and are especially prevalent during masting years when trees release heightened volumes of mast. Variations in masting can cause fluctuations in animal populations and can provide important information relating to weather patterns and other environmental factors such as climate change.

One of the main goals of the Duke Forest is to study masting cycles in order to better understand forest trends and to educate the public about the environment. To conduct this research, the Duke Forest staff collects thousands of mast to study. The original method of mast collection employed by Duke Forest researchers was a simple laundry basket with a net on top to separate out sticks and leaves. However, it takes Duke Forest researchers a significant amount of time to sort the collected mast by genus and to remove all other organic debris by hand for testing after collection. Thus, the Duke Forest researchers sought a design which would ultimately remove organic debris from the collected mast as well as sort the mast by genus.

In Fall of 2018, this design project was assigned to a group of first-year Duke engineering students in partnership with Duke University's First Year Design experience to give practical, real-world applicable tasks to incoming freshman engineers. The project was then continued in the Spring of 2019 by a different team.

Design

Before mast can be sorted by genus, they first need to be isolated from other natural debris. In pursuit of this goal, the device was designed with a clear focus on the efficacy of the collector: accurately filtering sticks, leaves, and other debris while still collecting all of the mast. Other factors were also considered, such as the cost, weight, and durability of the collector. The final mast collector design (Figures 3, 6) is split into two parts: an upper collector and a lower collector. The lower collector uses the relative shapes and sizes of different tree debris to sort them using a metal ramp inclined at a specific angle.

Upper Collector: To prevent larger branches from falling into the device and clogging up the lower collector, the upper filter was created using a PVC frame and a flexible mesh net (Figure 3). The goal of the upper filter is to block out larger debris that could possibly impede the collector's function by blocking the metal ramp while still allowing mast to pass through, allowing for the collector to operate for a longer time before requiring maintenance.

Figure 2: The upper collector is designed to block large debris from clogging up the lower collector.

Figure 3: The completed prototype.

Lower Collector: The lower collector, shown in Figure 5, is made out of a strong and lightweight polycarbonate material and held together by lightweight 3D printed plastic joints and metal L-brackets, which allows for ease of transportation and maneuverability, along with making the entire design modular. This component of the device separates mast from other tree debris using their kinetic energies. Since mast are more spherical in shape, they are able to roll down the ramp while other debris slides at a much slower speed. This difference in speed allows the mast to launch over the gap between the ramp and the collection bin while sticks and leaves fail to clear the gap. The unwanted debris then slides into the gap and out of the bottom of the mast collector onto the forest floor.

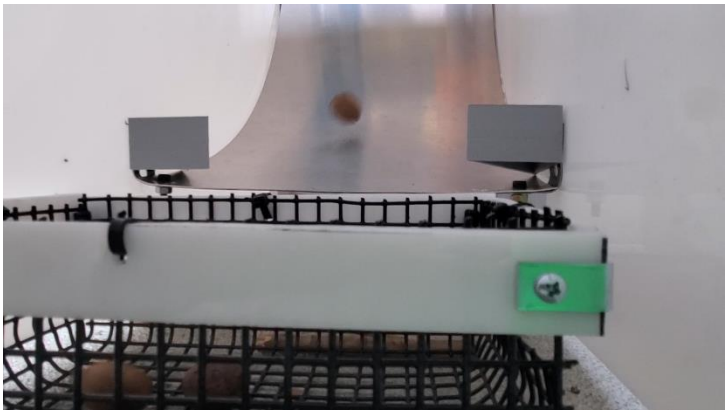

Figure 4: The lower collector separates mast from other debris using kinetic energy and a metal ramp.

To determine the optimal distance between the edge of the sorting ramp and the collection bin, a Python simulation was used to compare the predicted trajectories of mast and debris. The results of this simulation are shown in Figure 6. The source code can be found in Appendix A.

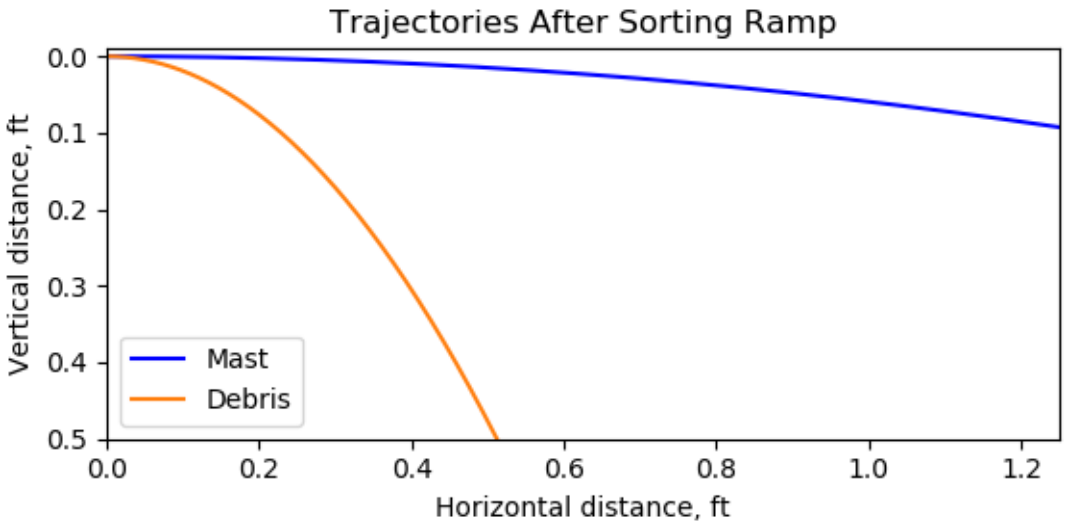

Figure 5: Predicted trajectories of mast and debris between the sorting ramp and collection bin.

Future Plans

Combined with a tarp system to funnel debris into the collector, the new collector is now small and lightweight enough to be easily transported and assembled, while still retaining the ability to separate mast from other natural foliage. By placing it near a grove of trees, collection of mast will become much easier. Now that mast can easily be isolated from other natural debris, the next step in the development of this project is to sort the collected mast by genus. This design challenge will be tasked to another group of freshman engineers at Duke in the coming Fall semester, and with any luck, the product will be ready for testing very soon.

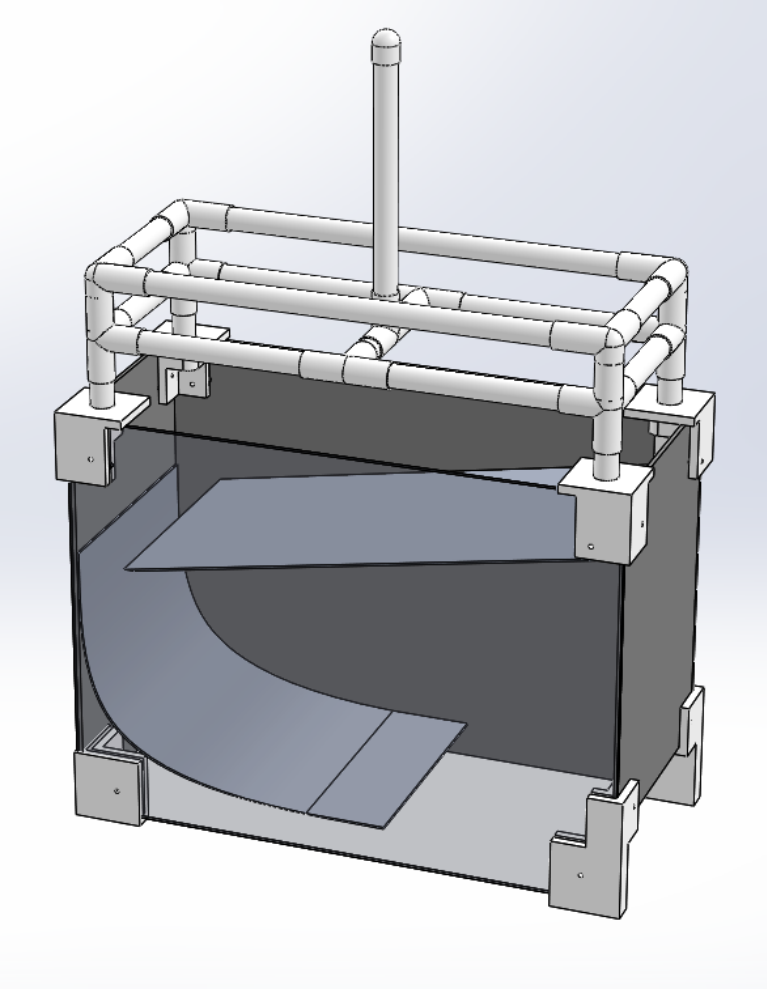

Figure 6: CAD model of final collector design.

Appendix A: Source Code

```
"""
Name: mast_sim.py
Created by: Thomas Howell

"""
#%% Imports
import numpy as np
import math as m
import matplotlib.pyplot as plt

#%% Define functions
def gen_trees(avg_height=15,std_dev=1,size=500):
    tree_heights = np.random.normal(avg_height,std_dev,size)
    return tree_heights

def drop_masts(tree_heights,length_1=1,theta_1=35*np.pi/180,height_2=1):
    v_f = []
    length_1 *= 0.0254
    height_2 *= 0.0254
    for k in tree_heights:
        if k>=0:
            v_f += [m.sqrt(2*9.81*(k+length_1*m.sin(theta_1)+height_2))]
    return np.array(v_f)

def drop_debris(mu,mass,length_1=1,theta_1=35*np.pi/180,height_2=1):
    v_f = []
    mass /= 1000
    length_1 *= 0.0254
    height_2 *= 0.0254
    if mass*mu<9.81:
        v_f = m.sqrt(2*(9.81-mass*mu)*(length_1*m.sin(theta_1)+height_2))
        return v_f

def trajectory(t,v):
    x = 3.28084 * v * t
    y = (-3.28084 * 4.905 * t**2) + 0.5
    return [x,y]

#%% Simulation
if __name__ == '__main__':

    #Generate data
    t = np.linspace(0,0.3,1000)
    v_mast = drop_masts([0],48,35*np.pi/180,23)
    v_debris = drop_debris(0.4,25)
    path_mast = trajectory(t,v_mast)
    path_debris = trajectory(t,v_debris)

    #Print values
    mast_loc = np.where((np.round(path_mast[1],4)==0))
    mast_dist = float(path_mast[0][mast_loc])
    mast_ft = m.floor(mast_dist)
    mast_in = round((mast_dist-mast_ft)*12)
    if mast_ft>0:
        print("Mast distance: {}'{}\"".format(mast_ft,mast_in))
    else:
        print('Mast distance: {}"'.format(mast_in))
```

```
58.
59.     debris_loc = np.where((np.round(path_debris[1],4)==0))
60.     debris_dist = float(path_debris[0][debris_loc])
61.     debris_ft = m.floor(debris_dist)
62.     debris_in = round((debris_dist-debris_ft)*12)
63.     if debris_ft>0:
64.         print("Debris distance: {}'{}\"".format(debris_ft,debris_in))
65.     else:
66.         print('Debris distance: {}"'.format(debris_in))
67.
68.     #Plot results
69.     plt.figure(1); plt.clf()
70.     plt.xlim(0,15/12); plt.ylim(-0.01,0.5); plt.title('Trajectories After Sorting Ramp')
71.     plt.xlabel('Horizontal distance, ft'); plt.ylabel('Vertical distance, ft')
72.     plt.gca().set_aspect('equal', adjustable='box'); plt.gca().invert_yaxis()
73.     plt.plot(path_mast[0],.5-path_mast[1],'b-', label='Mast')
74.     plt.plot(path_debris[0],.5-path_debris[1],'-',color='tab:orange',label='Debris')
75.     plt.legend(loc='best')
76.     plt.savefig('mast_trajectories.png')
```