Multi-threaded Space Carving for 3-D Seed Reconstruction

Mitchell L. Neilsen, Kai Zhao
Computer Science Department
Kansas State University
Manhattan, KS, USA
{neilsen@ksu.edu, kzhao@ksu.edu}

***Abstract* – This paper presents a novel framework for 3D seed reconstruction and volume measurement using multi-threaded space carving. The new framework achieves better performance than previous approaches with a relatively low cost. On modest hardware, the framework can compute seed volume within 0.5 seconds, but on a more modern machine with multiple cores, a new multi-threaded version can obtain a solution in 0.03 seconds and achieve nearly linear speedup for up to 6 threads. The new framework provides a practical, cost effective solution to accurately obtain seed volume within 1-2% even with very modest hardware.**

Keywords— Computer vision, cyber-physical systems, phenotyping, seed analysis, space carving.

I. Introduction

Plant breeding programs need to measure morphometric properties of plants and seeds, called phenotypes, to establish relationships between those properties and plant behavior – growth rate, yield, etc. Many plant and seed traits are measured for complex trait assessment (Li et. al., 2014). For seeds, an important trait is seed volume, but the estimation of seed volume is complex because many seeds have an irregular shape. A seed is the most basic agricultural entity from which complex root and shoot systems develop. Research on seeds is heavily dependent on seed volume, which is also used to compute density (Saatkamp et al., 2019). In seed bank studies, seeds are required to be sorted by size. Conventionally, seeds are sorted by passing them through a series of sieves of diminishing mesh sizes. The pitfall with this technique is that sieves only separate seeds according to a linear dimension, rather than by volume. In addition, it is a tedious and time-consuming task that may damage the seeds (Sankaran et al., 2016).

Other popular techniques to estimate seed volume include water displacement, volume slicing (Cao et al., 2020) and silhouette-based volume carving (Zhao et al., 2022). The standard technique of water displacement determines the volume of an object by computing the amount of water displaced when the object is submerged in the body of water. However, seeds float, absorb water, and subsequently bloat, so this technique is detrimental to the structural integrity of seeds. Volume slicing refers to the technique of dividing the object in an image into a fixed number of cross-sections and computing the volume of each of the cross-sections, which are assumed to be concentric (Cao et al., 2020). The sum of cross-section volumes is an estimate of total volume. Silhouette-based volume estimation is a technique of constructing a 3-D model of an object from multi-view imagery using a technique called space carving (Kutulakos et al., 2000) or volume carving. These are based on the seminal work of Martin to reconstruct a 3-D object from multiple views (Martin, et al., 1983).

While no volume estimation technique is 100% accurate, the image-based techniques of volume slicing and silhouette-based space carving preserve the structural integrity and chemical composition of the seeds, while providing a high level of accuracy. Another key benefit to image-based analysis is that the results are reproducible and eliminate subjective bias between experiments. The practical estimation of seed volume requires timely processing of seeds. We developed a low-cost, end-to-end silhouette-based space carving framework to reconstruct a 3-D model of a seed using multi-view imagery using a fixed camera perspective and rotating the seed as shown in Figure 1 (Zhao et al., 2022).

Figure 1. Hardware configuration

This paper extends the previous work to present a more efficient silhouette-based space carving framework using multiple threads The proposed system is low-throughput because it only processes one seed at a time, but it provides the accuracy needed to evaluate other novel high-throughput platforms which are currently under development.

II. Related Work

Koc (Koc, 2007) demonstrated the use of ellipsoid slicing and image processing techniques to estimate the volume of watermelon. Dimensions, including length, and major and minor diameters of the watermelon, are used as the parameters in the ellipsoid approximation method. A grayscale contour of the watermelon, called the boundary, is extracted using image

processing. From the boundary, volume is estimated using the standard disk method from Calculus (Riddle, 1979). The outcomes of the two techniques are compared to measures computed by water displacement. The comparison shows that image processing techniques perform better than ellipsoid slicing.

Roussel, Geiger, Fischbach, Jahnke and Scharr (Roussel et al., 2016) describe a method to perform 3-D reconstruction of plant seeds surfaces with diameters as small as 200 μm using the shape-from-silhouette approach. The robotized system developed as part of the work is low-throughput and processes one seed at a time. The system allows for camera pose variations – images captured from different angles.

Cao and Neilsen extended this work to produce a simple and affordable 3-D single seed volume measurement system (Cao et al., 2020). The method involves the use of 3-D printed components to act as the scaffolding onto which cameras are mounted. The seed is placed on a turntable with a black background. Images of the seed are captured by the camera as a seed rotates on a turntable. The captured images are processed using silhouette-based volume sculpting to estimate volume. The system was verified using a ceramic ball of known size, and the computed size is within 3% of the actual size. Zhao, et al., extended this result to produce a simple system with a blue-colored background or lighted background (Zhao, et al., 2022) as shown in Figure 1. The same hardware configuration is used for this paper.

Yang and Cho (Yang et al., 2021) performed 3-D crop reconstruction and automatic analysis of phenotyping index using machine learning. In the experiment, a system was configured and implemented for the 3-D image reconstruction of red pepper plant, as well as its automatic analysis. A Kinect v2 with a depth sensor and a high-resolution RGB camera were used to obtain more accurate reconstructed 3-D images. The reconstructed 3-D images were compared with conventional reconstructed images, and the data of the reconstructed images were analyzed with respect to their directly measured features and accuracy, such as leaf number, width, and plant height. The results showed that the proposed method showed an error of about 5 mm or less when reconstructing and analyzing 3-D images, and was suitable for phenotypic analysis.

Potmesil (Potmesil, 1987) presented a method to generate octree models for 3-D solid objects from their silhouettes obtained in a sequence of images. The silhouettes of objects were projected into an image and the center of projection generated 3-D conic volumes. The 3-D model of the objects is constructed by intersecting such conic volumes obtained from a sequence of images. To process 3-D volume data efficiently, hierarchical octree structures were used. The volumes of individual objects were labelled by a connectivity-labeling algorithm, and surface-normal vectors were added to their surface volume elements.

Sankaran, Wang and Vandemark (Sankaran et al., 2016) proposed an image-based rapid phenotyping technique to estimate the size of chickpeas. The experiment was conducted on samples collected from 72 plots from two different locations. The experiment involved the use of images where chickpeas were clustered together and separated from each other. The Watershed Algorithm was used to segment the clusters of chickpeas and a macro developed in ImageJ was used to estimate seed count and seed size. The reference object used to estimate chickpea size was the US Penny. The results exhibited a high correlation between the seed size estimated by the image processing technique and ground-truth data.

Golbach, Kootstra, Damjanovic, Otten and Zedde (Golbach et al., 2016) proposed a high-throughput system for 3-D reconstruction of plants. The 3-D reconstruction was based on the use of a silhouette-based technique. To make the process high-throughput, varying number of cameras were used to capture the image of the plants from different perspectives. While the quality increased with the increase in the number of cameras, it also led to an increase in the computational complexity of the system. In the end, ten cameras were chosen to balance the trade-off. The results indicated a high correlation between the results obtained by the algorithm and ground-truth measurements obtained by manual measurement.

III. Materials and Methods

The terms and concepts used to explain the estimation of volume using a silhouette-based approach are described below. The complete underlying mathematical derivations are beyond the scope of this article.

Voxel: A voxel is the unit of information that defines a small region in 3-D space. A pixel in 2-D space is analogous to a voxel in 3-D space. Simply, voxels are virtual cubic blocks that represent regions in 3-D space.

World Coordinate System: The world coordinate system is the basic 3-D Cartesian coordinate system with an origin assigned at a fixed point; it defines any point in 3-D space relative to the origin.

Camera Coordinate System: The camera coordinate system is a 3-D coordinate system relative to the camera's origin and orientation. It is possible to apply the operations of rotation and translation to convert a point in the world coordinate system to the camera coordinate system. In particular, we define a 4 x 4 transformation matrix known as Camera Extrinsic Matrix which applies rotation and translation to convert a point from the world coordinate system to the camera coordinate system. The camera extrinsic matrix changes with the physical location of the camera.

Image Coordinate System: The image coordinate system is the coordinate system that projects a 3-D point from camera coordinate system to a 2-D plane. Only the height and width of the 3-D point are captured and there is no sense of depth in the image coordinate system. Hence, it is a lossy transformation that does not have an inverse.

Pixel Coordinate System: The pixel coordinate system is used to discretize the image coordinate system which is divided into pixels. The pixel coordinates of an image are discrete values within a range computed by dividing the coordinates in the image coordinate system by pixel width and pixel height.

Intrinsic Parameters: The parameters that define the relationship between the image coordinates and camera

coordinates are called intrinsic parameters. Intrinsic parameters are specific to the camera in use and calibration of the intrinsic parameters is required to avoid image distortion. The intrinsic parameters typically considered are:

- **Focal Length**: Focal Length is defined as the distance between the image plane and origin of the camera coordinate system.
- **Principal Point**: The Principal Point is the point where the optical axis, i.e., the z-axis of the camera coordinate system, intersects the image.

A. Seed Reconstruction Shape-from-Silhouette Method

The silhouette of an object in an image refers to the contour that separates the object from the background. The shape from silhouette method requires multiple images captured from different angles to perform 3-D reconstruction of a seed kernel. For each of the images captured, the silhouettes are segmented using a method known as background subtraction. The silhouettes are then back-projected onto a common 3-D space with projection centers equal to the camera locations. The back-projection onto a common space requires the intrinsic camera matrix K and distance between the origin of the working volume and camera center. The origin of the working volume is set to be the intersection point (IP) of the seed bottom horizontal line and the line drawn through the center of the image. A total of N images equidistant from each other are spaced at rotation angles $a_i = i * 360/N$ for $i \in \{0\ldots, N-1\}$. If N=36, then images are spaced at 0°, 10°, 20°, .. , 350° as shown in Figure 2.

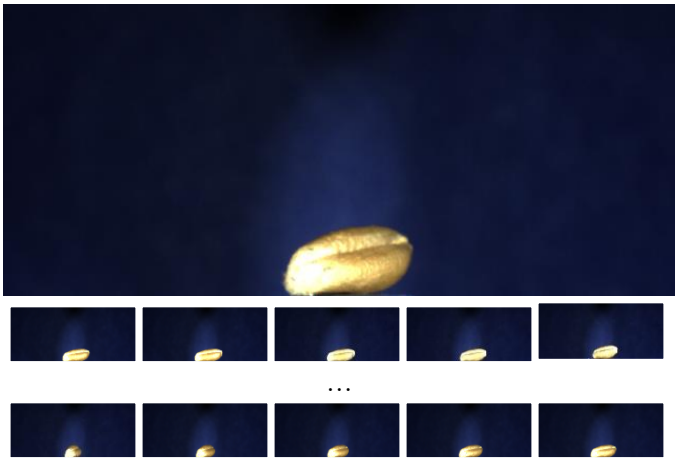

Figure 2. Example images captured by camera.

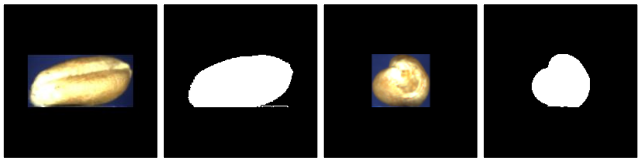

Figure 3. Identified regions of interest (RoI) and masks.

The rotation is around the vertical axis through the IP and parallel to the Y-axis of the camera. A grayscale threshold on each of the acquired images is generated and segmented into a binary mask M_i where $i \in \{0\ldots, N-1\}$.

For each image, the camera projection matrix P_i is calculated from the rotation angle a_i by $P_i = K(R_i \mid T_i)$ where R_i is the rotation matrix and T_i is the translation matrix corresponding to the given angle a_i. Then, an equidistantly spaced cubic voxel grid around the world origin is defined around the world origin. The size of each voxel is set to 1 mm^3. Each voxel center with homogenous world coordinates $\vec{X}$ is projected to a point, $\vec{x}_i$ in each mask M_i by $\vec{x}_i = P_i \vec{X}$. If a voxel belongs to the foreground object, its value V ($\vec{X}$) is set to 1. If the voxel does not belong to the foreground object, its value V ($\vec{X}$) is set to 0.

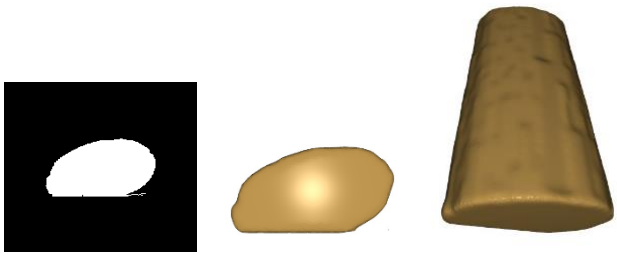

Figure 4. Mask, voxels after carving (front), high-angle view.

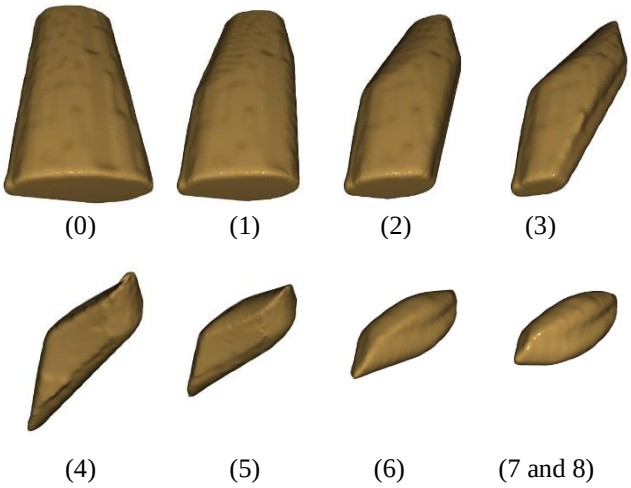

Figure 5. Space carving with masks 0, 1, 2, 3, 4, 5, 6, and 7/8.

The mask M_i in space carving is sensitive to misalignment of the object volume and needs to be adjusted precisely.

Figure 5 shows the results of carving the 10 mm x 10 mm x 10 mm cube voxel box sequentially with masks 1-10. After carving with just one mask, the volume is reduced from 1,000 mm^3 to 102.354 mm^3. About 90% of the voxels are carved away. After a second mask, rotated by 10 degrees is carved, the volume drops to 85.821 mm^3. And then to 69.544 after 3, 53.102 after 4, 37.953 after 5, 25.462 after 7 (masks 6 and 7),

and 22.371 after 9 (masks 8 and 9). After the first 35 masks are carved away, a volume of 20.478 remains. The final volume is 20.475 mm^3 as shown below in Figure 6, so only three voxels are removed by the final mask if they are applied sequentially to carve. If multiple threads are carving at the same time, there is no problem with write synchronization because threads only switch voxels from 1 to 0.

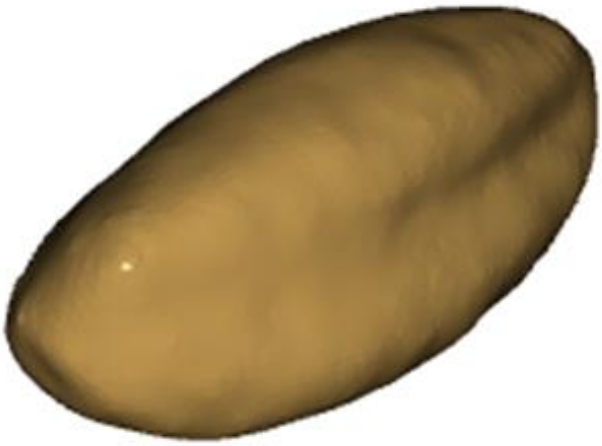

Figure 6. Final carved seed voxels.

After projecting a given voxel onto the mask, if the mask pixel is 0, then the voxel is set to 0. Thus, the same voxel could be set to 0 by several threads.

Input data is stored in a binary file, called carveInput.dat, with data in little-endian format. The file format is defined by:

- 12 bytes: voxel size in x-direction (sX), y-direction (sY), z-direction (sZ), stored in integers.
- sX * sY * sZ bytes: initial voxel values (typically all 1s, so they could be set internally to reduce the size of the input file); if sX = sY = sZ = 100, then 1 MB is used to store the initial voxel values. For debugging and to create the voxels resulting after each image is carved away, as shown in Figure 5, it is nice to have this additional flexibility.
- 4 bytes: number of masks to process (integer m). Repeat m times:
    - 96 bytes: projection matrix, P, for mask – 12 double-precision floating point values each
    - 8 bytes: mask size in x-direction (msX) and y-direction (msY); for our testing msX = msY = 200, stored as integers.
    - mask array, stored as msX x msY array, or 200x200 byte array in row-major order.

  If a full complement of 36 masks are stored, then 36 x 40,104 = 1,433,744 bytes are used to store masks and projection matrices.
- 24 bytes: physical width, height, and depth as doubles (10.0 mm each)

The maximum size is 1,000,016 + 1,443,744 + 24 = 2,443,784 bytes. As noted, we could easily reduce the size by 1 MB by not providing the initial voxel values. It would be interesting study the effect of storing voxels as bits instead of bytes. In the same way, masks could be stored in bits vs. bytes.

B. Hardware Configuration

The 3-D reconstruction of seeds requires multiple images to be captured from different angles at regular angle intervals. To ensure that the images are captured precisely, a low-cost hardware setup is proposed. Different parts of the hardware setup are described as follows and shown in Figure 1.

- **Seed Station:** The seed station shown in Figure 1 carries the seed as it is rotated by a NEMA 23 stepper motor. The seed station is just big enough to ensure that the seed comfortably fits on it. The stepper motor is powered by a 12V power supply and controlled by an Arduino microcontroller using a Synthetos gShield stepper driver.
- **Image Capture:** The images of the seed are captured using a color camera DFK 37BUX287 manufactured by ImagingSource IC. Two cameras that are placed orthogonal to each other were used in the original setup, but this configuration only uses a single camera. The image capture is automated using an API provided by ImagingSource IC software.
- **Stepper Motor:** The stepper motor is controlled by an Arduino UNO R3 board fitted with a Synthetos gShield v5 board. The stepper motor is controlled by G-code strings sent to it from a program written in Python. The turn rate in degrees is customizable by the user using micro stepping. For the experiment, a 10-degree rotation is used as the default setting to obtain 36 images in one complete rotation.

C. Software Framework

The software implementation to control the hardware framework and 3-D reconstruction is written in Python and C. The complete algorithm used to process the images, captured by the hardware shown in Figure 2, is as follows:

- Crop each image so that the Region of Interest (RoI) is at the center of the image as shown by the blue background in Figure 3.
- Convert the images from RGB color space to HSV color space, and color threshold the image to better separate the foreground (seed) from the background.
- Identify image contours – several different techniques can be used, and several more are being investigated. The Sobel operator can be used to perform a 2-D spatial gradient measurement on an image to highlight regions with high spatial frequency corresponding to edges.
- Apply binary thresholding or another technique to create a mask from each contour as shown in Figure 3.
- Apply a space carving algorithm using each of the binary masks and their corresponding projection matrices to construct a model of the seed and count the number of remaining voxels to estimate the seed volume. The total volume of 3-D reconstructed voxels is $\sum_{i=1}^{n} v_i$ where $v_i = 1$ x 1 x 1. The total volume of seed kernel is computed by multiplying by a factor which is the cubic volume of a pixel

per mm stored in camera properties. The space carving algorithm used is the focus of this paper.

- Finally, compute a point cloud of the model constructed for display of the seed models constructed as shown in Figures 4, 5, and 6.

The algorithm used for space carving is simple, and can be easily parallelized. For each mask, carve away the voxels that are not projected onto a mask (the white portion of the image shown in Figure 4). Thus, different masks can be processed in parallel by different threads, each responsible for processing a subset of the masks. Although it is logical to allow all of the threads to run in parallel, it turns out to be more efficient to have one (or possibly 2) threads run first and preprocess the voxels by carving away a single (or orthogonal) masks.

Suppose that the voxels are stored in a byte array V[depth][height][width], and the projection matrices are stored in P[mask][The basic algorithm is based on the classic Space Carving Algorithm (Kutulakos, et al., 2000), but instead of projecting each voxel onto all masks, we consider the voxels that can be carved away by projecting them onto a single mask. The voxel data structure is shared by all threads. Since they are only setting an initial voxel value from 1 to 0 if the voxel is carved away, there is no race condition.

The **Base Space Carving Algorithm** is shown below:

1. Initialize the voxels V to be a volume containing the true object. For the experiment, we just use a simple 3-D 100 x 100 x 100 voxel grid with all voxels set to 1.
2. For each mask:
   For each voxel $v \in V$:
   a. Project v onto the mask to 2-D point π_v
   b. If π_v is in the background of the mask (i.e., it is non-photo-consistent) (black) and v is 1, set $v = 0$

In this way, each mask can be processed by a different thread. However, once a voxel has been carved, it's not necessary to see if it should be carved away again. Thus, we can add a test to check the target voxel to see if it has already been carved, before going to the work of projecting it onto the mask to see if it should be carved based on the current mask. This leads to a much more efficient second algorithm, which is called the **Check Target Space Carving Algorithm**:

1. Initialize the voxels V to be a volume containing the true object. Set all voxels in the voxel grid to 1.
2. For each mask:
   For each voxel $v \in V$, **if v == 1:**
   a. Project v onto the mask to 2-D point π_v
   b. If π_v is in the background of the mask (i.e., it is non-photo-consistent) (black), set $v = 0$

As we shall see, this performs better for a small number of threads, but it actually performs worse for a large number of threads because of the extra "if" test. Finally, to really speed things up, we found that greater benefits can be realized by allowing just one (or a few) threads to pre-carve the object, before unleashing most of the worker threads to finish the job.

The intuition is that, due to caching, if we had 36 threads all carving based on their own mask, then they would need to check all voxels. In our case, that is 1,000,000 voxels. But, as we saw with a typical seed in Figure 5, about 90% of the voxels are removed by carving out with the first mask. Thus, we settled on a **Preprocess + Check Target Space Carving Algorithm**:

1. Initialize the voxels V to be a volume containing the true object. Just set all voxels to 1 as before.
2. **For one mask, preprocess the voxels:**
   For each voxel $v \in V$:
   a. Project v onto the mask to 2-D point π_v
   b. If π_v is in the background of the mask (i.e., it is non-photo-consistent) (black), set $v = 0$
3. For each **remaining** mask **using updated voxels V**:
   For each voxel $v \in V$, **if v == 1:**
   a. Project v onto the mask to 2-D point π_v
   b. If π_v is in the background of the mask (i.e., it is non-photo-consistent) (black), set $v = 0$

Of course, the improvement in speed depends on the fact that the object (in our case a seed) is a relatively small percentage of the total number of voxels in 3-D space; in our example, around 10%, but significant improvement in performance should be obtained with even a larger number of voxels remaining after the preprocessing step, at least up to 20%.

IV. Performance Analysis

Experiments using multiple seeds were performed on a Dell G16 7620 Windows 11 Pro with an Intel® Core ™ i7-12700H CPU @ 2.3 GHz, 14 Core(s), and 20 Logical Processors, and 16 GB RAM, and a more modest Dell Latitude 7480 Windows 10 Pro with an Intel® Core ™ i7-6600U CPU @ 2.6 GHz, 2 Core(s), and 4 Logical Processor(s), and 8 GB RAM.

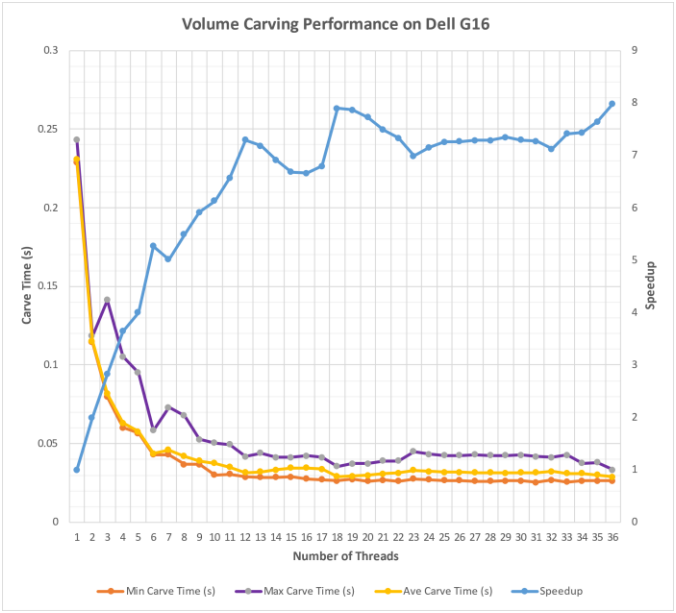

Figure 7. Performance analysis.

For the Base Space Carving Algorithm, running on a Dell G16, the average read time is **2.75** milliseconds. This could be reduced to 1.75 ms if we assumed the voxels would be set to all 1's initially. With one thread, space carving takes an average of **0.231** seconds as shown in Figure 7. This average was computed by running the experiment 500 times. The time ranged between 0.229 sec. to 0.243 sec. Nearly linear speedup is achieved going from 1 to 6 threads. With 6 threads, the speedup is **5.27** with an average carve time of **0.044** sec. After that, note that 7 threads exhibit worse performance because there are 36 tasks which are evenly divided among 6 threads, but not evenly divided among 7 threads. With 6 threads, each thread is assigned 6 tasks, but with 7 threads, most threads are assigned 5 tasks, but one thread is assigned 6 tasks. The next peak in performance is seen with 12 threads (each thread is assigned 3 tasks) and the speedup is 7.29. Next, with 18 threads (each thread is assigned 2 tasks), the speedup is 7.89 with a carve time of **0.029** sec. Finally, a slightly better speedup of 7.98 is obtained with 36 threads.

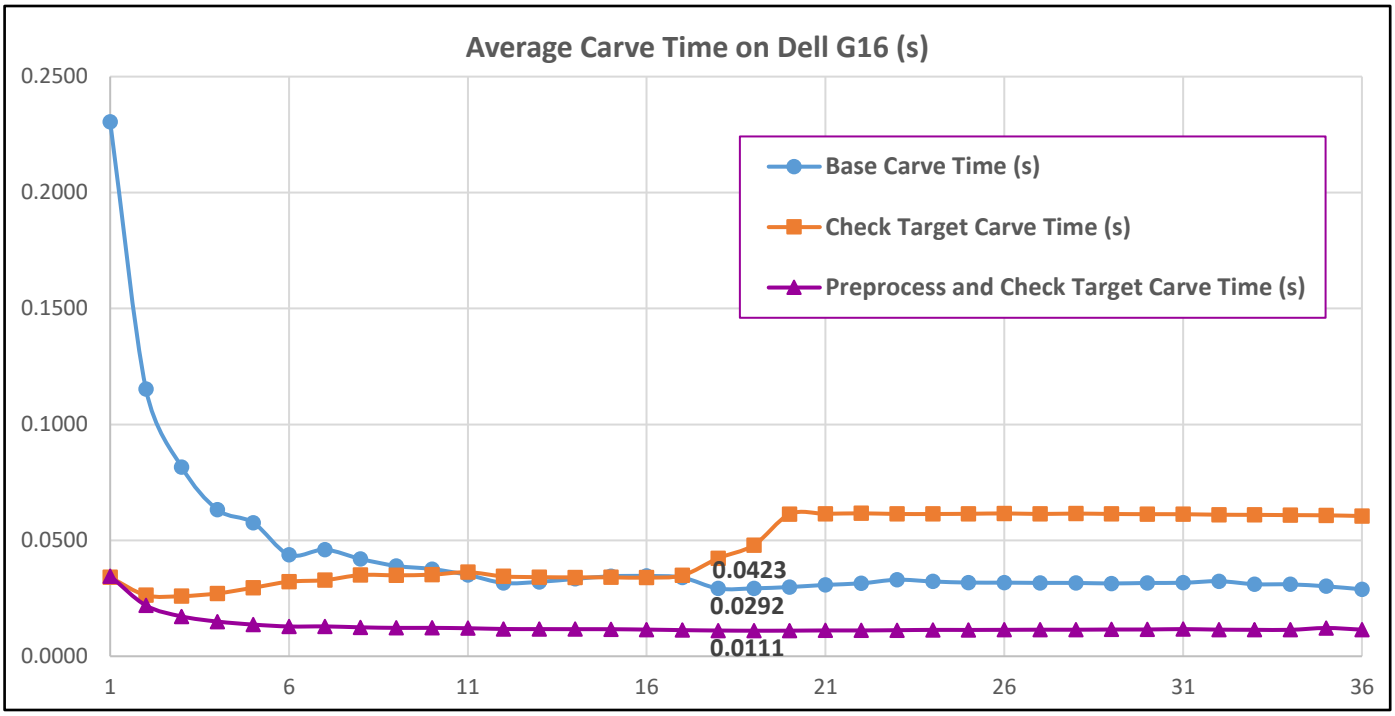

Figure 8. Performance analysis of all three algorithms on Dell G16

A very interesting result is obtained while comparing the execution time of all three carving algorithms on a Dell G16. We completed 500 trials for each algorithm using a range of worker threads, from 1 to 36, and compute the average carve time required for all threads to complete as shown in Figure 8.

Remaining tasks are assigned to threads using a simple modular assignment. With 36 tasks numbered 0 to 35 (each associated with one of the masks to carve), and with n threads, numbered 0 to n-1, in the first two algorithms, thread i is assigned tasks { i + j*n | j=0, .., k, where (36-i)/n -1 ≤ k < (36-i)/n }. For example, if n = 2, thread 0 is assigned all even tasks and thread 1 is assigned all odd tasks. If n = 36, then each thread is assigned one task; that is, thread i is assigned task i, for i = 0, .. , 35. For n = 12, each thread is assigned three tasks:

Thread 0: task 0, 12, 24
Thread 1: task 1, 13, 25
..
Thread 11: task 11, 23, 35

In this way, the threads can be scheduled to run on 12 cores, each with an equal amount of work For n = 18, each thread is assigned two tasks. After carving, we completed a sanity check to ensure that the same number of voxels remained. For our test case, 21,234 voxels of the available 1,000,000 remained after carving. And it only took about 0.0001 s = 0.1 msec. to compute. The time to write out the 1MB voxel cube took about 0.0006 s = 0.6 msec. For the Base Algorithm, the total time to read, volume carve, validate, and write the result was only about **0.033** seconds on average with 18 threads (of that 0.029 seconds for carving), and ranged up to **0.235** seconds for one thread (with 0.231 seconds for carving). Thus, in all cases, the dominant time was the time to complete space carving, accounting for over 87% of the time required.

To speed that up, we added the check to verify that a voxel hasn't already been carved. The second algorithm, Check Target, obtains an optimal speed of 0.0259 with just 2 threads. Note however that once the number of threads exceeds more than half the number of tasks (36), then the performance degrades to be worse than the Base Algorithm. This is because the additional test isn't really helping any more.

Finally, to obtain the best result, we execute one thread sequentially to preprocess or pre-carve the voxels before the

other threads even start running. The purple line in Figure 8 shows that this approach is best and achieves a carving time of 0.0111 seconds consistently over a wide range of threads.

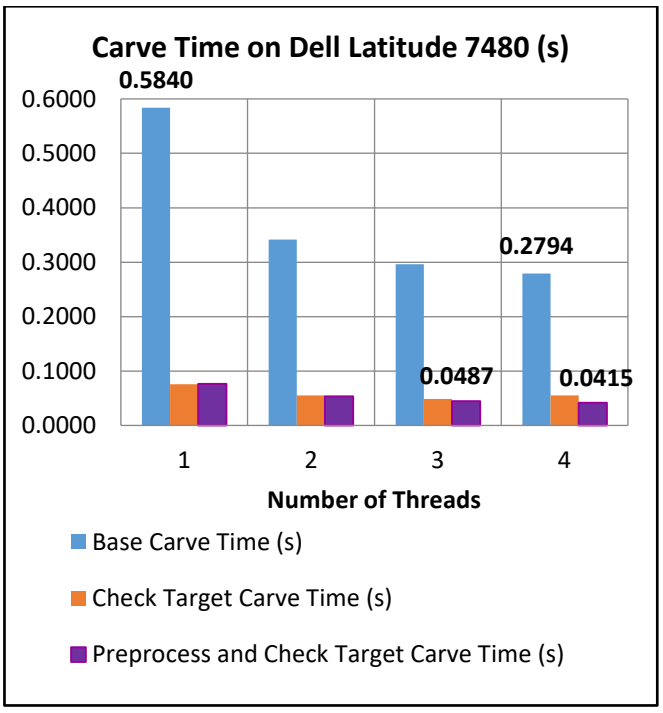

Figure 9. Performance on Dell Latitude Dual Core

With more modest hardware, similar results are obtained. The same test is performed on a Dell Latitude 7480 with 1 to 4 threads. Since the hardware only has a dual core, adding more threads doesn't improve the results. For the Base Algorithm, the best performance is obtained with 4 threads, with an average carving time of 0.2794 seconds. By checking the target, the second algorithm achieves an optimal carving time with three threads at 0.0487 seconds. Finally, preprocessing helps a little bit and achieves an optimal carving time of 0.0415 seconds with four threads. The improvement is not as much because fewer cores are available, but still better than the Base Algorithm.

V. Conclusions and Future Work

The proposed technique demonstrates an end-to-end 3-D reconstruction technique for volumetric estimation of seed kernels and its improved performance over the previous setup. In addition, the proposed technique also provides better results in comparison to volume slicing using ellipsoid approximation technique where images are captured using the two-camera and mirror-based setups. In upcoming iterations of the 3-D reconstruction setup, one of the ideas is to build a seed station with non-reflective material or to capture images with a lighted background. Since the current seed station holder is 3-D printed using a material that reflects light, it may appear in the captured images. Preliminary results indicate that using a lighted LED background improves the precision of measurement. Another direction for future research is to use GPUs to complete space carving. However, since the time required just using CPUs is relatively small compared with the time required to capture the images, the extra setup time may not be warranted. Finally, we plan to investigate the computation of volume using a point cloud derived from the remaining voxels. Currently, the volume is computed as the sum of the voxels remaining. To facilitate research, all algorithms, data, and results are available via GitHub.

References

[1] Li, L., Zhang, Q., & Huang, D. (2014). "A review of imaging techniques for plant phenotyping" . *Sensors*, *14*(11), 20078-20111.

[2] Saatkamp, A., Cochrane, A., Commander, L., Guja, L. K., Jimenez-Alfaro, B., Larson, J., ... & Walck, J. L. (2019). A research agenda for seed-trait functional ecology. *New Phytologist*, *221*(4), 1764-1775.

[3] Sankaran, S., Wang, M., & Vandemark, G. J. (2016). Image-based rapid phenotyping of chickpeas seed size. *Engineering in agriculture, environment and food*, *9*(1), 50-55.

[4] Cao, C. and Neilsen, M. (2020) "Efficient Seed Volume Measurement Framework," 2020 International Conference on Computational Science and Computational Intelligence (CSCI), pp. 1328-1334.

[5] Kutulakos, K.N., Seitz, S.M., "A theory of shape by space carving." Int'l Journal of Computer Vision, 38.3 (2000): 199-218.

[6] Martin, Worthy N, and J. K Aggarwal. "Volumetric Descriptions of Objects from Multiple Views." IEEE transactions on pattern analysis and machine intelligence PAMI-5.2 (1983): 150–158. Web.

[7] Zhao, K., Margapuri, V., and Neilsen, M. (2022). Automated phenotyping of single seeds using a novel volume sculpting framework. *ASABE Annual International Confeence*. Paper Number 2200403, Houston, TX, July 17-20, 2022.

[8] Golbach, F., Kootstra, G., Damjanovic, S., Otten, G., & van de Zedde, R. (2016). Validation of plant part measurements using a 3D reconstruction method suitable for high-throughput seedling phenotyping. *Machine Vision and Applications*, *27*(5), 663-680.

[9] Koc, Ali Bulent. "Determination of Watermelon Volume Using Ellipsoid Approximation and Image Processing." Postharvest biology and technology 45.3 (2007): 366–371.

[10] Riddle, D.F., 1979. Calculus and Analytic Geometry. Wadsworth Publishing Company, Inc. Belmont, CA, USA. pp. 506 – 507.

[11] Potmesil, Michael. "Generating Octree Models of 3D Objects from Their Silhouettes in a Sequence of Images." Computer vision, graphics, and image processing 40.1 (1987): 1–29. Web.

[12] Yang, Myongkyoon, and Seong-In Cho. "High-Resolution 3D Crop Reconstruction and Automatic Analysis of Phenotyping Index Using Machine Learning." Agriculture (Basel) 11.10 (2021): 1010–.

[13] Roussel, Johanna et al. "3D Surface Reconstruction of Plant Seeds by Volume Carving: Performance and Accuracies." Frontiers in plant science 7 (2016): 745–745.

[14] Cao, C. (2020). Computer vision frameworks for physics-based simulation of liquids and solids, Ph.D. Dissertation, Kansas State University.