# RASCAL: Robotic Arm for Sherds and Ceramics Automated Locomotion

Deborah Wang[1], Brandon Lutz[1], Peter J. Cobb[2], and Philip Dames[1]

***Abstract*— Ceramics are one of the major sources of information about the past for archaeologists, with a typical archaeological dig unearthing 1000's of pottery fragments (sherds) each day. However, archaeologists often are not allowed to remove these sherds from their home countries. Therefore, logging data (*e.g.,* mass, color, decoration) in the field is the only way to record valuable information about these sherds. Currently, this laborious process is done manually, using up much of the valuable time at a dig site. This project aims to automate the data collection process, freeing up archaeologists to spend their limited time in the field on other tasks, and to create a large-scale digital database of sherd information, allowing archaeologists to take advantage of new computational tools to make discoveries. The contribution of this paper is an automated system, consisting of several reconfigurable data collection stations and a robotic arm to transport objects between stations, that can rapidly generate a large database of archaeological artifacts. We validate our system in simulation, using high-resolution models of sherds. In other contexts, our system may help to expand the use of automation in materials handling, parts sorting, and more.**

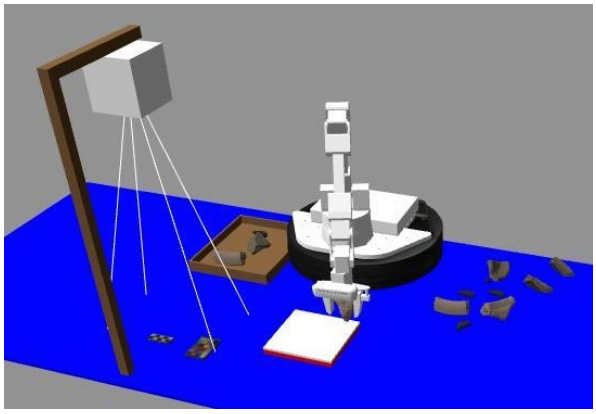


Fig. 1. Our automated data capture system, which includes a robotic arm, two reconfigurable data collection stations (digital scale and archival camera with color standard and scale bars), and designated pickup and dropoff areas.

## I. Introduction

Archaeological excavations and surface surveys produce large amounts of data about the past, with pottery being the most common material class uncovered by excavation. This abundance of pottery is because of its durability and because it was used for everything from the transport and storage of goods, to the preparation and serving of food and drink. As a result, archaeologists rely heavily on ceramic evidence to understand the human past. Research in this field depends on having accurate information about pottery manufacturing processes, surface colors and ornamentation, shape and related function, and archaeological context (*i.e.,* where it was found and what else was nearby). Currently, most archaeologists manually record these data in single-site databases, illustrate pottery by hand, and share the results only in paper publications, limiting large-scale discovery.

Our long-term goal is to accelerate archaeological research by bringing the collection, sharing, and analysis of ceramics data into the digital age by using a robotic system to autonomously create a database of ceramic sherds. These data, along with the context (*i.e.,* find location) of each sherd, can help us to advance our understanding of multiple past processes. For example, by puzzling sherds back into whole vessels we may trace the geological and erosional processes that have moved sherds around the site. We can also trace stylistic variance between sites and over time by matching similarly shaped or decorated fragments from different vessels or sites. In summary, this would be truly transformational for archaeology, revolutionizing our ability to study the past by applying cutting-edge big data analytics.

The use of digital reconstruction technology has rapidly expanded in recent years, both in the context of archaeology [1] and beyond, with one of the key challenges being the creation of digital models [2]. Much of the work in this area focuses on large-scale scene reconstruction, whether that be of entire archaeological sites [3]–[6] or of modern manufacturing facilities [7], [8]. These reconstructions are often created using sensor data gathered by a teleoperated or autonomous mobile robot. At the level of individual objects, there is much less existing work and most of it focuses on the data processing backend, attempting to automate the pipeline from raw sensor data to digital model in both archaeology [9] and manufacturing [10], [11]. However, none of these efforts, working at the level of individual objects, have attempted to automate the process of data capture, as we do in this paper. Data capture has been automated in other aspects of robotic manipulation, such as for object shape reconstruction for grasp planning [12]–[14]. However, the resulting models do not have other metadata associated with them, which is essential for archaeological studies.

The primary contribution of this paper is to develop a robotic system, which we call RASCAL (Robotic Arm

*This work was supported by the Temple University Presidential Humanities and Arts Research Program and by Facebook AI Research.

[1]Deborah Wang, Brandon Lutz, and Philip Dames are with the Department of Mechanical Engineering, Temple University, Philadelphia, PA, USA {deborah.wang, brandon.lutz, pdames}@temple.edu

[2]Peter J. Cobb is with the Faculties of Education and Arts, University of Hong Kong, Hong Kong pcobb@hku.hk

Fig. 2. A typical dig site from Peter Cobb's field work in Armenia during the summer of 2019.

for Sherds and Ceramics Automated Locomotion), to autonomously create a database of pottery sherds. While we focus on the application to archaeology, we designed RASCAL to be easily adaptable to new data collection needs. This will allow the results of our project to extend beyond the context of archaeology. For example, large online marketplaces sell millions of distinct products, with many new products added every day, making manual data collection difficult. Our automated system could be used to generate a library of digital models of the products stocked in distribution centers. These models could be used to determine the optimal shipping container and packing configuration to fulfill a customer order [15]. Similarly, in brick-and-mortar retail our system could create a database of products to lower the barrier to automating both the unpacking of boxes and the stocking of shelves by pre-identifying pickup points for each object [16].

## II. Archaeological Background

The purpose of archaeological field work is to discover physical evidence from the past and carefully document information about each individual object that is discovered. This information is then used to infer knowledge about the past, including during analysis that takes place later, away from the archaeological site. It is critical to stress that *all* data collection must happen in the field, as foreign archaeologists cannot ship materials back home due to international regulations restricting the export of historical artifacts from their countries of origin. This means that, during fieldwork, archaeologists must carefully balance time spent searching for artifacts versus cataloging data about artifacts, as there is no point in gathering additional artifacts if there is not time to collect information about them.

Figure 2 shows a team of researchers working at a typical dig site where they can unearth anywhere from 500 to 2000 pottery sherds in a day, the vast majority of which have a mass under 250 g and are less than 10 cm wide. Sherds are then placed carefully into bags or bins with other artifacts from the same *context*, *i.e.,* the place where an artifact is found, including information such as the soil type and color, the site type, the layer the artifact came from, what else was in that layer, etc. After a long day of digging, the excavators take the bags of pottery to a local field lab where each artifact is cleaned. Researchers then spend many hours recording multiple types of information on each object, such as the mass, color, and decoration. Given that it takes roughly 2 min to manually record the data of a single sherd, this amounts to 17-67 person-hours every day. This is clearly a very time-consuming, repetitive, and dull process. It is our goal to automate this portion of the fieldwork, returning to archaeologists hours of time each day that they can use to dig for additional artifacts, perform some preliminary data analyses in the field to better plan out the next day's activities, or to rest.

In the environments where Peter Cobb and his team work, the post-dig processing is usually done on standard folding tables set up in an open-sided structure, such as a tent or garage, located miles away from the dig site. This is done so that the researchers will have electricity to power laptops and other equipment, and to provide lighting after the sun sets. The working conditions in these structures are dusty and illumination is inconsistent, *e.g.,* due to passing clouds and the sun setting.

## III. System Design

The main elements of our automated data collection system, shown in Fig. 1, include: a robotic arm, reconfigurable data collection stations, a task-level planner, and a machine perception system. The remainder of this section details each of these components.

### A. Software

We use the Robot Operating System (ROS) [17] to develop the back end of our system, such as the motion planning for the arm, sensor interfaces, etc. We control the robotic arm through PyRobot [18], an open source Python-based interface written on top of ROS. We use PyRobot for two main reasons. First, PyRobot is a Python library with a simple interface that was specifically designed to lower the barrier to entry for robotics. In our context, this will enable archaeologists to more easily use the system and to debug any problems in the field. Second, PyRobot provides a standardized, hardware-independent interface. This allows our overall data collection system to seamlessly transition to other arms and to be used for other data collection purposes.

### B. Robotic Arm

We use the 6 degree-of-freedom WidowX 250 Mobile Robot Arm from Trossen Robotics. This arm offers full 360 degree rotation at the base and has a maximum reach of 68 cm, allowing it to operate over the majority of a standard folding table (the typical field setup we expect to encounter). The working payload is 250 g, which is greater than the vast majority of sherds unearthed at a dig site. Additionally, the WidowX 250 already has a PyRobot interface, as it is part of the LoCoBot[1], a platform natively supported in the PyRobot library. We modified the arm slightly, placing an Intel RealSense D435 depth camera on the wrist. This

[1] `http://www.locobot.org/`

Fig. 3. The simulation uses highly detailed models of actual sherds collected during past archaeological fieldwork.

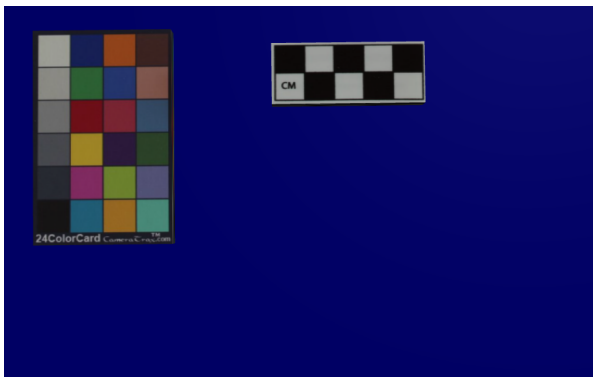



Fig. 4. The archival camera station displaying the color standard and scale bar. This is an image taken from the archival camera's point of view.

provides the camera with a clear line of sight to the table and allows the arm to easily detect sherds for pickup.

### C. Simulation

Due to COVID-19 restrictions, we have been unable to perform hardware testing. As a result, we developed and validated our system using Gazebo [19]. To minimize the gap between simulation and the real-world environment, we used high-definition digital models of actual sherds collected during past archaeological fieldwork, as shown in Fig. 3, and varied the lighting conditions.

### D. Data Collection Stations

For our particular application, we use two data collection stations: a digital scale to collect the mass of each sherd and a camera with color standard and scale bar (shown in Fig. 4) to collect an archival photo of each sherd. These data are entered into a PostgreSQL database, which also contains other metadata such as the context of that sherd.

*1) Digital Scale:* The digital scale is used to collect the mass of each sherd. This is important as the mass, in conjunction with the size, provides information about the density and, therefore, the material composition of a sherd. In our simulation, we implement this as a contact sensor, returning the mass of the sherd model that is placed on the scale. When moving to hardware, we plan to use a scale with a USB serial interface so that RASCAL can autonomously send tare commands and receive measurements.

*2) Archival Camera:* The archival camera is a downward-facing, high-definition camera mounted on a stand above the working surface together with a color standard and a scale bar. This station is used to collect the true color and overall

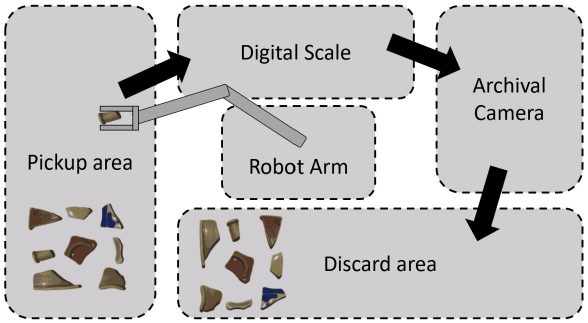



Fig. 5. Flow chart of RASCAL's data collection process.

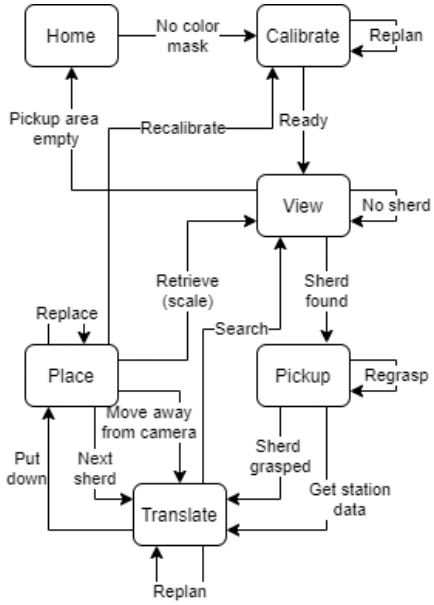



Fig. 6. The logic of RASCAL's state machine.

size. In our simulation, we created photorealistic models of the color standard and scale bar, shown in Fig. 4, and use the Gazebo camera plugin to simulate the camera images. In hardware, Peter Cobb uses a Canon EOS M200 camera. This is a compact, mirrorless CMOS camera with a wifi interface. Previously, Peter Cobb integrated this camera into a custom Android app used for manual data collection. We will leverage this to create an interface for RASCAL.

### E. Task-Level Planning

Figure 5 shows the workflow for our data collection procedure. The sherds all start in the Pickup Area, then the arm moves them through the two data collection stations (first the Digital Scale, then the Archival Camera) before placing them in a Discard Area. At each stage sherds are deposited at a consistent location on the working surface. The accompanying video shows RASCAL stepping through this workflow for five sherds.

We use the SMACH library [20] to implement this workflow on the arm. SMACH is a ROS-independent Python library that creates composable task-level workflows in the form of finite state machines. This provides a flexible

framework that includes specifications on transitions between stations and the ability to recover from failure. Figure 6 shows the logic in our state machine, which has six states to complete the necessary tasks: *Home*, *Calibrate*, *View*, *Translate*, *Pickup*, and *Place*. Outcomes and transitions can be quickly added or modified in SMACH, making the state machine robust against both predictable and unforeseen failure scenarios. For example, to pick up a sherd RASCAL attempts to fully close the fingers, allowing it to detect failures and drops by checking the finger positions. The accompanying video shows RASCAL re-locating and recovering a dropped sherd before moving on to the next station.

The combined use of PyRobot and SMACH allows users to quickly define the data collection workflow for their particular applications. This makes RASCAL easy to adapt to new hardware, new contexts, and new types of metadata. The source code is available on GitHub at `https://github.com/TempleRAIL/Archeology-Robot-Arm`. The remainder of this subsection outlines the procedure that RASCAL follows at each station.

*1) Initialization:* To initialize the system, RASCAL begins in the Home state. Next, it proceeds through a sensor calibration routine (see Sec. III-F for more details).

*2) Pickup Area:* The first stage in the data collection workflow is to acquire a sherd for processing. Unprocessed sherds are placed within a designated region on the table to one side of RASCAL. The arm sweeps the wrist-mounted camera over this region, following a lawnmower pattern with the camera pointed straight down, in search of sherds (View state). The parameters of this lawnmower pattern can be easily modified by the user, who can set both the height above the working surface and the step size in a configuration file. When RASCAL detects a sherd (see Sec. III-F for details of the perception system), it will stop the lawnmower pattern and then attempt to pick up the sherd (Pickup state). If no sherds are detected after scanning the whole pickup region, RASCAL will return home (Home state).

There are only two requirement for users. First, they must input the context ID for this bag of sherds so that RASCAL can correctly input the data into the database (note: sherds from the same context bag are simply numbered sequentially in the database). Second, they must place the sherds within the pickup area such that each sherd is separated from its neighbors by at least the width of the robot's fingers. The size of the Pickup Area depends on the reach of the robotic arm in use. Likewise, sherd geometry limits the maximum number of sherds that can fit into the pickup area. Our simulation allows 6-12 sherds to be placed within the pickup region, though the sherd models we use are relatively large. In practice, we will expand this area by decreasing the size of the base of the arm and further optimizing the mat layout to maximize the number of sherds that can be processed without any human intervention. Future work will improve the perception system (Sec. III-F) to segment overlapping sherds, allowing RASCAL to process full bins of shards without human intervention.

*3) Digital Scale:* The second stage in the data collection workflow is to record the mass of the sherd. Moving clockwise, the arm carries a sherd acquired from the pickup area to the digital scale (Translate state). The gripper always places the sherd at the center of the scale (Place state). After a short time to allow the sherd to settle, RASCAL records the mass measurement before attempting to retrieve the sherd from the same location where it was dropped off (Pickup state). If this first attempt fails, RASCAL moves to a survey position above the scale and uses the perception system (View state) to re-locate the sherd.

*4) Archival Camera:* The third stage in the data collection workflow is to take an archival photograph of the sherd. Continuing clockwise, the arm carries the sherd from the scale to the region below the archival camera (Translate state). As at the scale, the gripper always places the sherd under the camera at the same position on the working surface (Place state), in this case near to the color standard and scale bar. Next, RASCAL moves to a standby position, out of view of the archival camera (Translate state), before capturing the image. The arm then moves to a specified camera survey position to re-locate the sherd (View state) before retrieving it (Pickup state).

*5) Discard Area:* The fourth and final stage in the data collection workflow is to place the processed sherd into the discard area. The user can modify the dimensions of this discard area in a configuration file. In our simulation, the arm carries the sherd from the archival camera to a collection box (Translate state), finishing on the opposite side of RASCAL from the pickup area. The sherd is deposited at a random location within the bounds of the box (Place state) before the arm returns to the Pickup Area to begin again.

*6) System Configuration:* All of the key parameters for RASCAL are set in a single YAML configuration file. These include overall system parameters, such as the length of the gripper, along with the layout of each station (*e.g.,* location relative to the base of the arm and size). The only tool required to set up the parameters at a new site is a ruler, making the system very flexible and easy to use.

### *F. Perception*

RASCAL uses a wrist-mounted Intel RealSense D435 RGBD camera for sherd detection. To detect sherds, we apply standard color masking techniques in the HSV (Hue-Saturation-Value) colorspace using OpenCV [21]. Specifically, we mask out any pixels in the image that fall within a specified range of HS values (we ignore the V values as these are most sensitive to lighting conditions). This provides us with a lighting-invariant background subtraction method, which is crucial for use in outdoor environments where the ambient illumination can rapidly change. Additionally, we use a blue working surface to provide a strong contrast with sherds [22], as blue is unlikely to dominate the color of ancient ceramics. We tested the robustness of this solution using a variety of lighting conditions (Fig. 7).

*1) Calibration:* We generate the color masks during the Initialization step, where the robot visits each data station

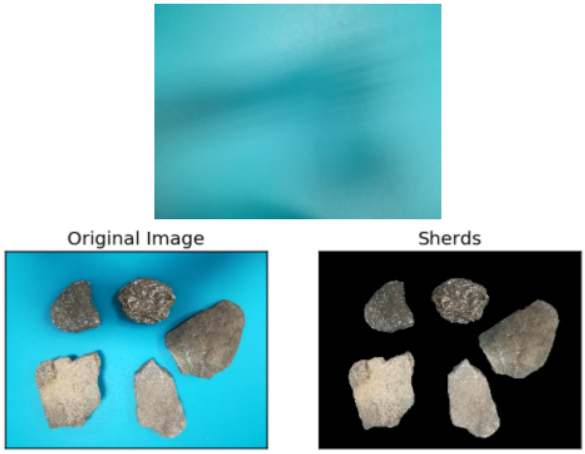



Fig. 7. To show lighting invariance, a color mask was generated from a working surface under low-light conditions (top), then applied to sherds on the same working surface under brighter lighting (bottom L) for successful segmentation (bottom R).

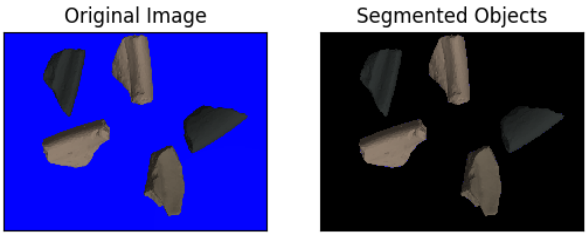
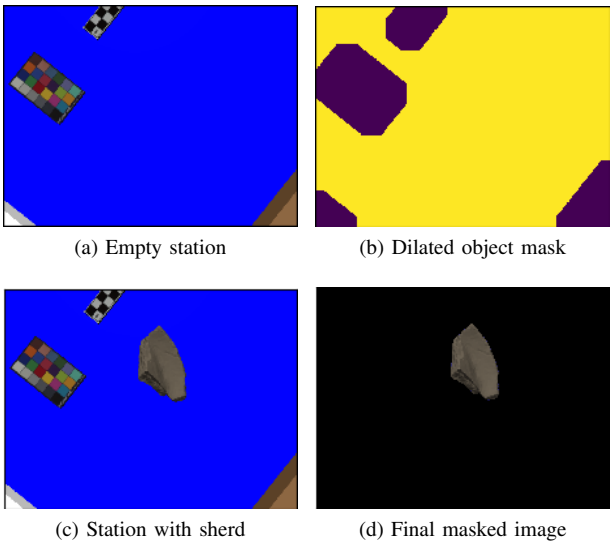



Fig. 8. Sherds in the pickup area before (L) and after (R) color segmentation.

prior to any sherds being placed there. At each station, the robot takes a picture and uses k-means clustering to extract a user-specified number of HSV values from this empty background image. These color masks use a fairly tight bound on hue values and a fairly loose bound on saturation values. Using k-means allows the module to handle multi-tone backgrounds (*e.g.,* white and blue at the scale station) or heavy shadows on a uniform background (Fig. 7). We periodically refresh these color masks to account for variations in illumination during operation.

*2) Image Segmentation:* At the pickup area and scale station, sherds are segmented by masking out all background-colored pixels, as seen in Fig. 8. At the archival camera station, the robot uses an additional step to mask out the color standard and scale bars. To create this mask, RASCAL takes a picture of the station with no sherd (Fig. 9a) and masks out the background color, resulting in just the color standard and scale bar. We then dilate this image (Fig. 9b), increasing the size of the mask around each object, to account for slight variations in the position of the arm at the survey location (see Sec. III-E). Then, both the color mask and the object mask are used to process images at the camera station (Fig. 9c) to yield estimates of the sherd only (Fig. 9d).

*3) Sherd Extraction:* After this segmentation process, we use OpenCV's `minAreaRect` function to extract the oriented bounding box of each sherd. This is used to determine



(a) Empty station (b) Dilated object mask

(c) Station with sherd (d) Final masked image

Fig. 9. Images showing the data processing steps at the camera station.

the pickup orientation for that sherd, so that the gripper pinches the sherd at the center along the minor axis. To determine the size of the sherd, we apply the same masks to the point cloud (aligned with the RGB camera frame). Heights for grasping and placing are calculated relative to the working surface, which is located at $z = 0$. Sherds not completely in-frame are filtered out to prevent the arm from attempting to grasp partial sherds.

### G. Sim-to-Real Transfer

We have designed our simulated system with sim-to-real transfer as one of the foremost considerations. We specifically selected the programming tools PyRobot and ROS [17] because they offer a consistent interface between both simulation and hardware. In addition, our use of the HSV colorspace will allow RASCAL to operate in outdoor environments with variable lighting conditions.

We believe the rendering of realistic sherd models, color standards, and scale bars has allowed us to account for many real-world variables in our virtual environment, which will ensure a smooth transition to the real world. The realistic sherd models allowed us to test the grasping capabilities of RASCAL on objects of comparable complexity to actual sherds uncovered at an archaeological site. The inclusion of the color standard and scale bar at the archival camera station allowed us solve the problem of handling foreign multicolor objects in the line of sight of the RASCAL camera, so that only sherds were detected as appropriate objects to include in RASCAL's workflow.

## IV. Results

RASCAL's primary purpose is to offload from archaeologists as many dull and repetitive tasks as possible. To do this, the system must work quickly and reliably. As a result, our primary metric of interest is the processing time per sherd.

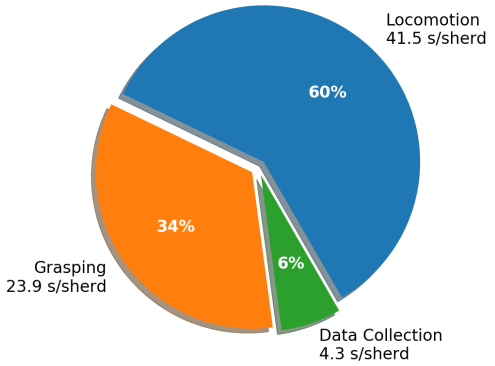



Fig. 10. RASCAL's processing time per sherd by subroutine, averaged over twenty sherds (two batches of ten).

### A. System Parameters

During testing, RASCAL used a working height (the nominal height above the working surface in the Translate state) of 18 cm and a survey height (nominal height in the View state) of 28 cm. To prevent the gripper from colliding with the working surface during grasping or placing, we included a small clearance of 0.25 cm.

Due to significant jittering of the fingers (at least partially caused by simulation artifacts during collision checking due to the high level of detail in the sherd models), we used the Gazebo grasp fix plugin to stabilize grasping behavior by quickly initializing the grip and slowly breaking contact [23]. We set the plugin's update rate to 50 Hz, the grip count threshold to 2, the maximum grip count to 100, the angle tolerance between forces to 100°, the release tolerance to 0.1 cm, and disabled collisions on attachment.

### B. Test Results

To test RASCAL we processed batches of 10 sherds, the approximate number of sherds that fit in the current Pickup Area. Excluding the one-time Calibration step, the average batch took a total of 698 s to complete, which comes to 69.8 s per sherd or 1240 sherds per 24 hours. This is right in the center of the total number of sherds excavated during a typical day (500-2000). Additionally, RASCAL is able to process sherds nearly twice as fast as a human (who has to manually enter data into the database, trigger the camera, etc.). Based on these results, our system has the potential to save approximately 41 person-hours per day, freeing up many of the team members to work on other tasks.

In addition to the total time, we also analyzed the time spent on different subroutines, shown in Fig. 10. *Locomotion* and *Grasping* times include all motion with and without a sherd in hand, respectively. *Data Collection* times include the motion to and from the standby positions while recording the mass and archival image. The *Calibration* step is not shown in the pie chart, as it only occurs once at the start of the batch. This procedure takes 6.0 s on average, or 0.9% of the total time for each batch.

Grasp failure, where RASCAL either fails to pick up a sherd or drops a sherd mid-motion, was the most common error that slowed down our system, with an average of 0.8 drop events per sherd. The vast majority of these drops were due to the aforementioned simulation artifacts in Gazebo, even with the use of the grasp fix plugin. See the accompanying video for an example of a drop event (due to jittering in the fingers) and the resulting successful recovery behavior. We anticipate the number of drops will decrease substantially during hardware experiments, saving around 5 s per sherd.

All of the data above include planning time. We use two different inverse kinematic solvers: MoveIt [24] when descending the last few centimeters to grasp a sherd or lifting a sherd the first few centimeters off the working surface, as these steps require the most careful motion, and PyRobot's built-in analytical inverse kinematics (IK) solver for the remainder of the motions, as this is significantly faster than MoveIt and yields qualitatively similar motions.

## V. Conclusion

In this paper we describe the design and evaluation of an automated data collection system for archaeological pottery sherds. At a typical dig site, archaeologists can unearth 500-2000 sherds per day. Currently, all data about this vast quantity of sherds is manually recorded, a dull and repetitive task that, at around 2 min/sherd, can take anywhere from 17-67 person-hours for each day's worth of digging. Furthermore, all data collection must take place in the field due to legal restrictions on transporting sherds out of their home countries. This means that all of the time spent cataloging data is time not spent searching for or analyzing artifacts.

RASCAL automates the data collection process, reducing the total time spent per sherd by nearly a factor of two and freeing up archaeologists to focus on more important tasks. Our system consists of multiple data collection stations and a robotic arm to transport sherds between them. The arm has a built-in camera and employs a perception module that works even during rapid illumination changes that may occur outdoors (*e.g.,* due to passing clouds). RASCAL has been designed to be easily set up at a new work site, requiring only a ruler to measure the locations of the data collection stations relative to the arm. RASCAL uses a finite state machine to step through the tasks, employing robust error checking and delivering repeatable performance. Furthermore, the use of PyRobot to control the arm provides a simple and hardware-independent interface, allowing non-roboticists to debug issues in the field and allowing the workflow to seamlessly transition to new arms and data collection needs.

We validated our system in a Gazebo simulation environment using detailed 3D reconstructions of data collection stations and of actual sherds. The results showed that RASCAL can process sherds at an average rate of 69.8 s/sherd, or 1260 sherds/day. This is nearly twice as fast per sherd as the estimated time for a human to collect the data, and, assuming that RASCAL can operate over a full 24 hour period, it can save a team of archaeologists 41 person-hours of time per day. Next summer we hope to validate RASCAL at a dig site with Peter Cobb and his team.

## REFERENCES


[1] C. H. Roosevelt, P. Cobb, E. Moss, B. R. Olson, and S. Ünlüsoy, "Excavation is destruction digitization: advances in archaeological practice," *Journal of Field Archaeology*, vol. 40, no. 3, pp. 325–346, 2015.

[2] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1016–1022, 2018.

[3] B. Bingham, B. Foley, H. Singh, R. Camilli, K. Delaporta, R. Eustice, A. Mallios, D. Mindell, C. Roman, and D. Sakellariou, "Robotic tools for deep water archaeology: Surveying an ancient shipwreck with an autonomous underwater vehicle," *Journal of Field Robotics*, vol. 27, no. 6, pp. 702–717, 2010.

[4] M. Johnson-Roberson, M. Bryson, A. Friedman, O. Pizarro, G. Troni, P. Ozog, and J. C. Henderson, "High-resolution underwater robotic vision-based mapping and three-dimensional reconstruction for archaeology," *Journal of Field Robotics*, vol. 34, no. 4, pp. 625–643, 2017.

[5] J. Fernández-Hernandez, D. González-Aguilera, P. Rodríguez-Gonzálvez, and J. Mancera-Taboada, "Image-based modelling from unmanned aerial vehicle (uav) photogrammetry: an effective, low-cost tool for archaeological applications," *Archaeometry*, vol. 57, no. 1, pp. 128–145, 2015.

[6] F. Remondino, L. Barazzetti, F. Nex, M. Scaioni, and D. Sarazzi, "Uav photogrammetry for mapping and 3d modeling–current status and future perspectives," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. 1, p. C22, 2011.

[7] B. Denkena, M.-A. Dittrich, S. Stobrawa, and J. Stjepandić, "Efficient generation of a digital twin using object detection for data acquisition and xml-interface for model creation," *Procedia CIRP*, vol. 93, pp. 274–279, 2020.

[8] V. Stojanovic, M. Trapp, R. Richter, B. Hagedorn, and J. Döllner, "Towards the generation of digital twins for facility management based on 3d point clouds," in *Proceeding of the 34th Annual ARCOM Conference*, 2018, pp. 270–279.

[9] A. Kai-Browne, K. Kohlmeyer, J. Gonnella, T. Bremer, S. Brandhorst, F. Balda, S. Plesch, and D. Lehmann, "3d acquisition, processing and visualization of archaeological artifacts," in *Euro-Mediterranean Conference*. Springer, 2016, pp. 397–408.

[10] M. Sommer, J. Stjepandić, S. Stobrawa, and M. von Soden, "Automatic generation of digital twin based on scanning and object recognition," in *ASIM Fachtagung Simulation in Poduktion und Logistik 2019*, 2019.

[11] S. Haag and R. Anderl, "Automated generation of as-manufactured geometric representations for digital twins using step," *Procedia CIRP*, vol. 84, pp. 1082–1087, 2019.

[12] A. Kurenkov, J. Ji, A. Garg, V. Mehta, J. Gwak, C. Choy, and S. Savarese, "Deformnet: Free-form deformation network for 3d shape reconstruction from a single image," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2018, pp. 858–866.

[13] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis, "Single image 3d object detection and pose estimation for grasping," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3936–3943.

[14] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1–6.

[15] F. Wang and K. Hauser, "Robot packing with known items and nondeterministic arrival order," *IEEE Transactions on Automation Science and Engineering*, 2020.

[16] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *2003 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 2003, pp. 1824–1829.

[17] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[18] A. Murali, T. Chen, K. V. Alwala, D. Gandhi, L. Pinto, S. Gupta, and A. Gupta, "Pyrobot: An open-source robotics framework for research and benchmarking," *arXiv preprint arXiv:1906.08236*, 2019.

[19] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3. IEEE, 2004, pp. 2149–2154.

[20] J. Bohren, "smach," Mar 2018. [Online]. Available: http://wiki.ros.org/smach

[21] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[22] A. R. Smith and J. F. Blinn, "Blue screen matting," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96, 1996, pp. 259–268. [Online]. Available: https://doi-org.libproxy.temple.edu/10.1145/237170.237263

[23] J. Buehler, "The gazebo grasp fix plugin," Mar 2016. [Online]. Available: https://github.com/jenniferBuehler/gazebo-pkgs/wiki/The-Gazebo-grasp-fix-plugin

[24] D. Coleman, I. A. Şucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a MoveIt! case study," *Journal of Software Engineering for Robotics*, vol. 5, no. 1, pp. 3–16, 2014.