PALISSY SNAKE PROJECT

A modern update to a historic artistic process.

Christopher Chinowth & Erica Sugimoto

06.14.2017

UCSD CSE 145 - Embedded Systems Design Project

ABSTRACT

French ceramicist, Bernard Palissy, was known for making casts of live animals to create large decorative platters. Our project is to create a modern spin on Palissy-ware. We want a humane way to capture the form and movement of an animal using today's technology. The problem is that there is no existing technology for capturing this kind of data, and the alternatives are inauthentic and compromises too much of the art.

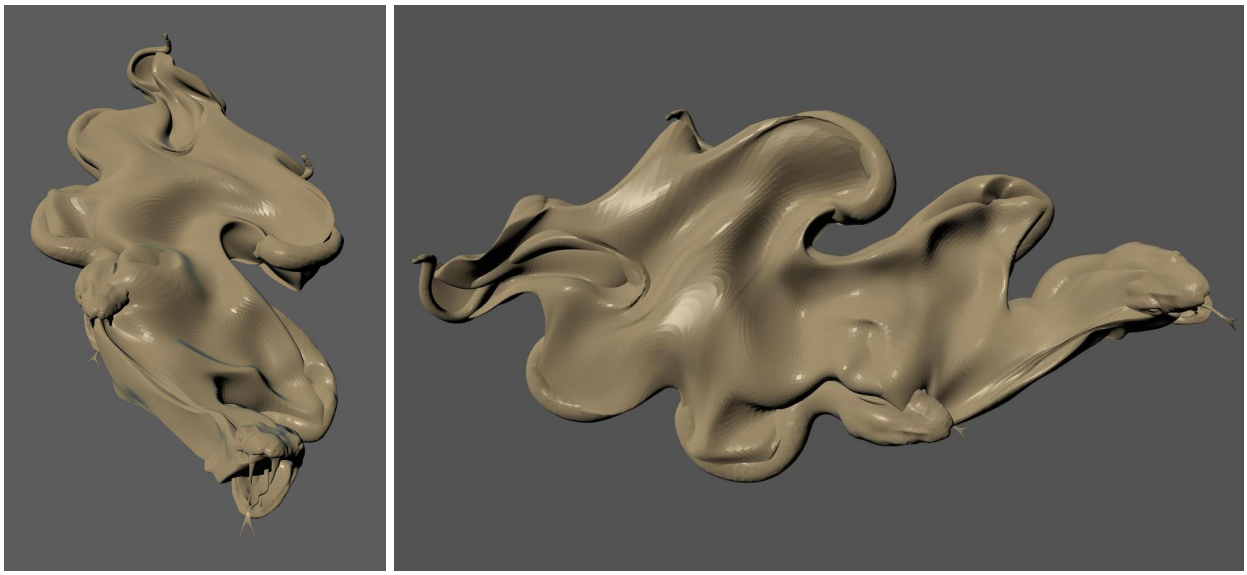
Our solution uses 3D cameras to create a live scan of the animal over time and use that data to create a 3D model. In the end, we have successfully created a brand new artistic medium that captures the aesthetic of movement in live animals.

INTRODUCTION

Bernard Palissy (c. 1510 - c. 1589) was a french ceramicist who created a technique called live casting to create incredibly detailed and realistic large oval platters featuring small animals and vegetation. His technique involved capturing live specimens in the marshes around his home. He would capture local frogs, snakes, snails, shells, and crayfish as well as plants around their habitats. He took extra care to not damage the animals as he captured them and refused to use already dead or maimed creatures, as they would not retain their true forms. (Hanna Rose Shell, 2004) He would then kill the animals by submerging them in vessels of urine and vinegar for half an hour. Once they were dead he would pose them in naturalistic positions upon a sheet of clay. Once they were in position and pinned to the clay he would then cover them in butter or olive oil to facilitate the making of a series of master molds out of plaster. (Francesco Pellizzi, 2014) These molds were then used to create the rusticware that he was famous for. He also used the molds to build a private grotto for Catherine de Medicis, the queen of France, at the garden of Tuileries palace. He would attach the ceramic forms of lizards, snakes, and shellfish to the walls of the grotto and glaze them with runny lead-based glaze to increase their watery realism. He built several grottos such as this one in his lifetime, but they have all crumbled over time such that very little remains of them. (Tom Turner, 2004)

Miljohn Ruperto, an LA artist who received a B.A. from University of California, Berkeley and an M.F.A. from Yale, came up with a concept that was inspired by Bernard Palissy's live casting. He wanted to do a prolonged 3D scan of a living animal, specifically a snake. The scan would last for roughly a minute as the snake slithers around an

enclosed area. Then the recording would be split up into intervals of time, possibly 10 seconds long, and the “frames” would be collapsed or fused together to capture the movement of the snake during the interval into one shape and form in a 3D model. Then these 3D models would be printed to create casts that will then be used to create porcelain sculptures that will be put on display together. He emphasizes that the piece not only showcases the form and movement of the snake but also the limitations of the technology used to create it, so any form of digital artifacts from the scanning would be acceptable. Below is a crude approximation of what Miljohn imagined the end product to look like.



The issue with this interesting and technical problem that Miljohn needed to be solved is that there are no ready made solutions for it. No one has done anything quite like this before so a new solution needed to be engineered for it. To that end, we have created a reproducible process for creating these 3D models using Intel RealSense technology which we break down into steps and go into detail below.

CAPTURE RAW SNAKE DEPTH DATA

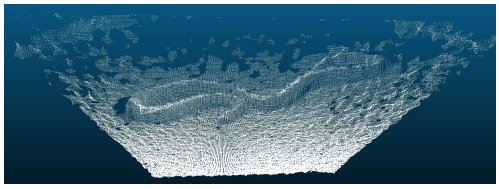
For the snake record, a snake handler was hired to bring in three Florida King Snakes for us to scan. Florida King Snakes are nonpoisonous constrictors, around four to five feet in length. We created a twenty-five square foot marked area in our carpeted lab for the snake to move around in. We referred to this area as the stage. When the snake moved out of the stage the snake handler would gently step in to move

the snake back to the center of the area. In the four corners of the stage, we set up Intel RealSense 3D cameras. Three of the corners had a ZR300 and one of the corners had an R200 set up on tripods. These 3D cameras implement stereovision depth imaging through an infrared projector and two infrared sensitive lenses. Both of these models have a depth stream resolution of 480 x 360 @ 60fps. We recorded the frame data in binary files containing rasterized 172,800 16-bit integers per frame. We also saved an additional file containing the camera specific intrinsic parameters which are used when processing the data into point clouds. We positioned our cameras in three different ways. One shot with the cameras on the corners of the stage. Another with the cameras two feet away from the corners. A third with the cameras four feet away from the corners. In all, we recorded about five hundred gigabytes of depth data.

SYNCHRONIZE FRAMES FROM REALSENSE CAMERAS

The four different cameras do not all start recording at the same time and they run on different clocks, so without some preparation it would be nearly impossible to synchronize the frame data of all four perspectives. Before every recording of the snake, we had someone hold a book in view of all four cameras. When the cameras all started recording he would close the book which we would use as a signal to synchronize the timestamps of the frames. That way we could simply go through the very beginning of the data, find the first frame for each camera where the book was closed and use that as the start of our data.

CREATE POINT CLOUDS FROM DEPTH DATA

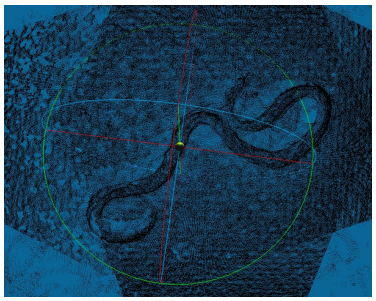


For converting our binary depth data into point clouds, we used The Point Cloud Library (PCL). PCL is a standalone, large scale, open project for 3D image and point cloud processing. First we would read in the data from our binary files and then convert each pixel value into a data point inserted into a point cloud object. Then using the appropriate camera intrinsics we would de-rasterize the data points so that they matched their actual measurements. We then saved these point clouds as PLY files using the PCL API. A PLY file is a file format specifically designed to store three-dimensional data from 3D scanners and is a commonly used file extension for other programs that we used. It was at this point that we realized that the configuration with cameras set up right on top of the corners of the stage gave us the best looking data for our purpose, so we began moving forward with specifically that data.

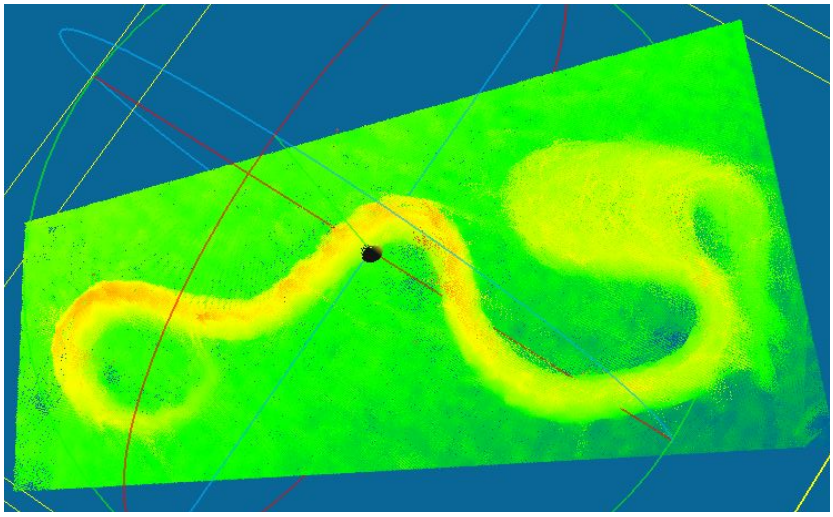
ALIGN POINT CLOUDS

Now we have synchronized point clouds for all of our frames and four different perspectives of the same scene. The problem here is that these four different perspectives are not aligned, meaning that they do not know exactly how to fit together to create a whole image. To solve this issue, for every new configuration of cameras we set up, we also set up a unique scene on the stage using a handful of ping pong and tennis balls. This scene looks unique from every camera angle, so we are able to create point clouds of this scene and then take them into a program called CloudCompare where we manually rotated them to fit together. CloudCompare then gives a transformation matrix for each perspective. By applying this transformation matrix to the data of the appropriate perspective, the data is automatically transformed and rotated to match with the other perspectives.

COMBINE POINT CLOUDS

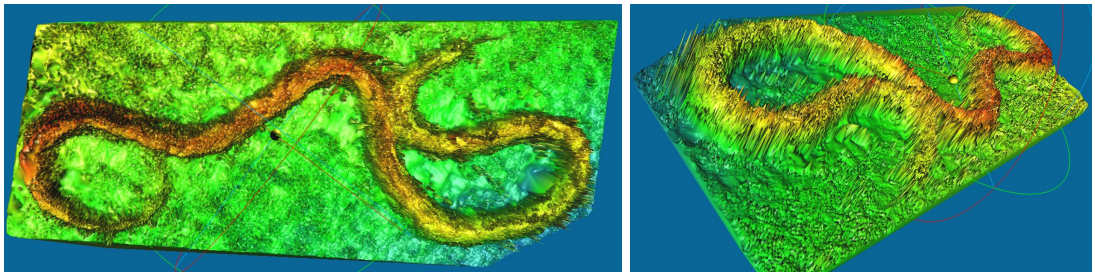


Now that we have aligned the point clouds we can save the point clouds together. The result creates a full image of the snake if we only combine four perspectives of one frame. In order to capture the movement of the snake in the point cloud, we simply combine the point clouds of the four perspectives of say six-hundred frames for a representation of a ten second interval shown below.

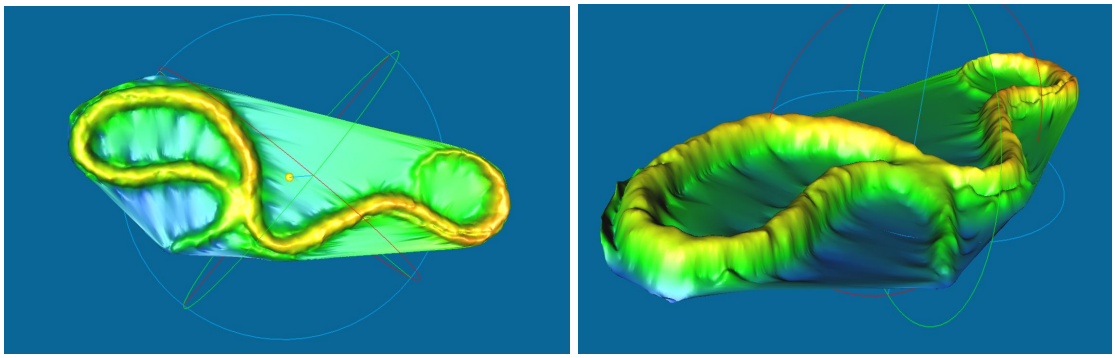


CREATE AND SMOOTH A MESH

Now that we have a combined point cloud we are ready to create a mesh. For this we used CloudCompare again using the Delaunay 2.5D (best fit plane) function. This function attempts to fit planes to all of the vertices in the point cloud as best it can, producing the following results.



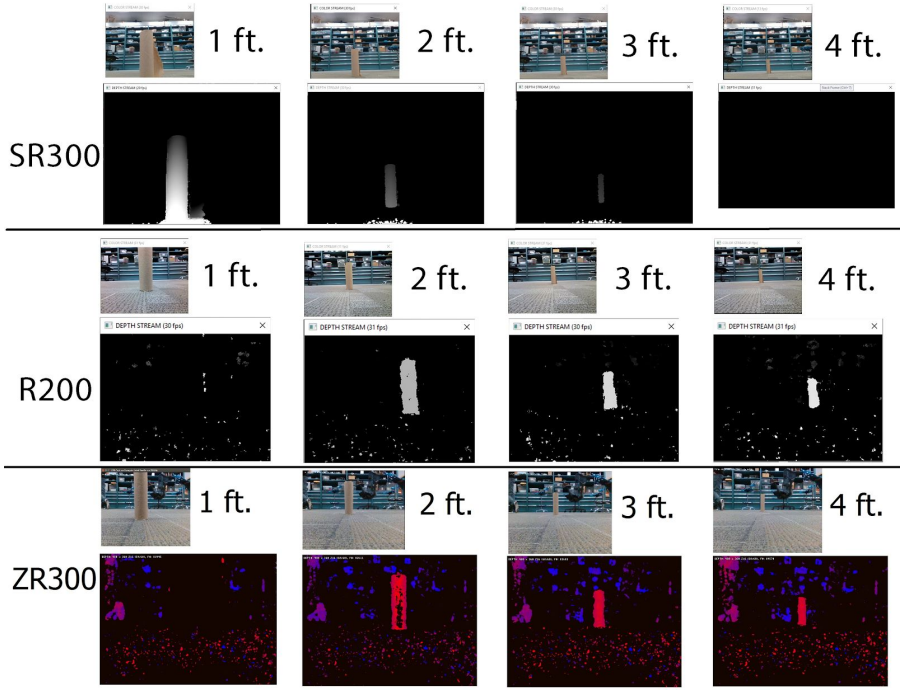
Now the mesh in its current state cannot be physically printed because of the steep peaks and valleys present all over the model. In order to alleviate this issue we move on to filter and smooth the model. For this purpose we used Laplacian smoothing to produce a model like this.



When we finally have models that we are satisfied with, we export them as OBJ files which are simple data-formats that represent 3D geometry. This is a common file extension for 3D printers and many other 3D graphics applications. These models will be printed and then used to create casts, which will then be used to create porcelain sculptures.

MILESTONES

- **Test and Research Cameras + Camera Selection**
  We had three different Intel RealSense cameras that we needed to choose between that would best work for our project. The R200 had a low resolution and was more suitable for recordings from greater than two feet away. The SR300 had a higher resolution and was particularly good at short range recordings, within two feet. The ZR300 had a high resolution and was good for longer range recordings much like the R200. We ultimately went with the ZR300 because it was like an upgraded version of the R200 and we needed to be able to record from distances of greater than two feet. A problem we faced here was that the ZR300s were not immediately available for us to test. We had to wait for them to come in before we could finalize our decision. While we were waiting for the ZR300s to arrive, we temporarily used the R200 in order to get started on the project. Another issue we ran into was that the SR300 would not work on our linux systems. We could get it to work properly using Windows but not linux. This basically eliminated the SR300 from our viable choices, in addition to the low range. We completed this milestone and created this interesting graphic comparing the quality of the cameras at different distances.



- **Setup Laptop**
  The RealSense cameras required USB 3.0 ports to plug into, which we did not have access to at the time, so we had to order a new computer. When it arrived we had to install linux and the librealsense library. However; we ran into a ton of problems. First, the laptop that was ordered had such new components that the linux drivers did not work well for it. Several installations we tried had broken graphics drivers and broken ethernet and wireless drivers. We also needed a kernel within a very specific and small range of versions in order to use the librealsense library because librealsense requires you to run a patch on your kernel in order to use the RealSense camera drivers. This entire process was a very long and arduous nightmare, but eventually was completed. We had to settle on an installation with a broken wireless driver and for some reason the laptop had really bad performance issues, but at the very least it was working.
- **Write a program to capture depth data**
  While we were waiting for the ZR300s to arrive we were able to successfully write a program that captures and records depth data from a RealSense camera. We validated this data was accurate by writing another program that visualized the data.
- **Improve the depth data capture program**
  We were able to improve the depth data capture program in several ways, we added a button to start and stop recording, we added a visual cue so that you could know when the program is recording depth data, we also modified it to create folders for each recording with time stamps to help us organize the data.
- **Camera Setup**
  We came up with many designs for potential sets for the cameras. Some options were to create an actual enclosure built out of plywood for the snake to slither around in. However; we were informed that the snake did not need to have an enclosure and that the snake handler would be able to guide it on an open area of floor. So our setup was turned into four cameras on tripods in all cardinal directions. We had various setups that changed the distance and the angles of the cameras so that we could get as varied data as possible. We ran into problems trying to run four RealSense cameras on the same computer though. At first we only had three USB 3.0 ports to work with, and the computer handled three RealSense cameras just fine. So we ordered a USB 3.0 hub and some extension cords. Turns out that the RealSense cameras consume a lot of power from the USB 3.0 ports and they were not able to output enough power for all four cameras. This created a ton of confusion for us and took us awhile to figure out. The only

solution for this would be to use multiple laptops running our software for the recordings. This created a new milestone to set up an additional computer.

- **Set up second laptop**
  We ended up borrowing a laptop that Eric Lo was using for another project temporarily. We had to set it up to be nearly identical to the first laptop and it would be responsible for two of the RealSense cameras. This went much smoother than the first laptop since we already had experience with getting the environment ready and all the nuances it required. However it still took several hours to set up completely.
- **Camera Setup Tests**
  We tested the camera setup with various objects including a rubber snake purchased from amazon. We hoped that the rubber snake would be as close as we could get to an actual snake for testing purposes. The rubber snake turned out to be quite small and a very hard test subject. But after testing we were convinced that if we could get any meaningful data out of our small rubber snake then getting data for the actual snake would be much easier and higher quality.
- **Pet Store Snake Test**
  We had planned to try testing our cameras on actual snakes before the snake handler arrived in order to confirm that the cameras would indeed pick up the snake. We were worried about the potentially glossy texture of the snake not showing up in our depth data because we had some trouble with some glossy objects that we used to test in the lab. This was somewhat unfeasible for us to complete and we didn't do it, but thankfully the snake showed up in our data just fine.
- **Write a program for creating point clouds**
  We created a program that would take in a depth data file that we recorded using our capture software that we wrote and convert it into a point cloud. For this we had to de-rasterize the depth data using the camera's intrinsic parameters in order to reconstruct the point cloud data that was unwarped and in real measurements.
- **Capture Live Snake Data**
  At this point we were ready to capture live snake data, we scheduled a time for the snake handler and Miljohn to come in. For the most part we had a very successful shoot, except that we learned that only three of the five ZR300 cameras we had were working properly. We remedied this situation by replacing the fourth camera with an R200. The resolution was slightly worse which meant less data points on the point cloud from the R200's perspective, but the result was

negligible and we still got great shots. We tried recording with several different configurations of camera, varying the distance from the stage. Unfortunately, when we started converting the data to point clouds we found that the snake blended into the noise of the ground with the configurations where the cameras were further away. So we really only got to use the data from the first close up configuration. We expected something like this to happen though and this was precisely the reason we recorded with several different set ups.

- **Modify the Point Cloud Program for Multiple Cameras**
  We modified our Point Cloud Program to take a start frame for each camera's data, a number of frames to convert to point clouds, and a step value to skip frames if we wanted. These choices were done using a config file that the program would open for its parameters.
- **Fuse Multiple Point Clouds Together**
  We used a program called CloudCompare to rotate and combine the point clouds from all four perspectives and frames. We ended up having to do this manually the first time when aligning the point clouds because the automatic functions that CloudCompare offers were not working for our data sets. This was a bit of a headache but we only had to do it once and then we could save the transformation matrix for use any time.
- **Convert Point Cloud Into a Mesh**
  We created a mesh from the point cloud using CloudCompare, we tried using another program recommended to us called MeshLab but the resulting meshes were not snake like at all.
- **Smooth the Mesh**
  Again, using CloudCompare we were able to apply various smoothing filters to the meshes. MeshLab also has the ability to apply smoothing filters to a mesh and has a lot more options so we might try taking a mesh from CloudCompare and filtering in MeshLab, but we have not done that yet.
- **Convert the Mesh into a 3D Model & export as .obj File**
  This was a very simple milestone after the mesh was created and smoothed, a simple export function from the toolbar in CloudCompare was used.
- **3D Print .obj File**
  We did not do this, it was not a necessity for us to 3D print the object ourselves as the ceramics residency in Italy would be doing that.
- **Adjust & Optimize filters**
  This is something we are still in the process of doing, figuring out what the best meshing and smoothing function are for the form we are making. For now it

seems that the best approach is Delaunay triangulation for meshing and laplacian smoothing.

- **Create a User Interface**
  We are trying to streamline our process and make it as automatic as possible. This is something we have not yet completed but plan to in order to pump out as many of these meshes as possible so that Miljohn can hand select what he wants to use.

CONCLUSION

In summary, we have successfully created a new artistic process that blends the cutting edge technology and engineering of today with an age old ceramics technique. With this process we can create pieces that offer new perspectives on shapes and forms that change through time but is also reflective and self aware of its own history. We designed and engineered a solution for something that no one has done before which is novel and unique. The sculptures will be put on display at the House of World Cultures in Berlin in 2018, which will bring attention to the technology that was used to create it and also create excitement for both engineers and artists. We would like to think that our project bridges the gap between engineering and the arts by creating this interesting cross section of art history and modern computer science. Miljohn is also interested in recreating the lost Palissy grottos. Bringing back the grottos that have been destroyed would be significant to art history and the process created for this project can be reproduced for this task as well.

REFERENCES

1. Castin Life, Recasting Experience: Bernard Palissy's Occupation between Maker and Nature by Hanna Rose Shell
2. Res: Anthropology and Aesthetics, 65/66: 2014/2015 by Francesco Pellizzi
3. Garden History Reference Encyclopedia: Historic books etc on garden design and landscape architecture by Tom Turner