Real Time Point Cloud Search, Upload, and Update

John Lore (jlore) and Richard Zhu (rzhu1) - November 17th, 2017

A shared point cloud of the real world is paramount to allow digital content to interact with reality and for many people to share in these experiences. Our project seeks to leverage parallel mobile computing to generate a point cloud via mobile camera, match that data to part of the existing point cloud, upload changes, and push updates to others in real time.

Project Website

NOTE: This project has pivoted from the original proposal to make an anti-parallel crytocurrency.

After in depth research in accomplishing our original goal of making a cryptocurrency, we found that all our realistic goals had been done extensively, and our grander plans were out of scope for this project. Our motivation shifted to create a mobile cryptocurrency. This led us to the idea of using location in addition to mobile camera & compute to mine the real world's data and create a 3D map i.e. pointcloud of the world. Our initial motivation would be to treat discovering a change in the global point cloud analogously to a crytocurrency transaction. Miners would then be alerted to scan the area as a verification of the change, resulting in an update to our blockchain-like pointcloud. Each would be awarded some coin according to a function of traffic in the area, time since the area was last updated, and physical size of the change found. However, we quickly realized that to create a cryptocurrency this way would first require design of the system to store a massive point cloud, localize a user, search the point cloud to match the user's view, identify changes, upload changes, and push those changes to others. We were compelled by this idea, and decided to run with it, maintaining our initial motivation to create a cryptocurrency. The deprecated initial project idea can be seen below.

Initial Project Proposal

Background

A point cloud is simply a set of data points, here specifically pertaining to data points in 3D space that describe the real world -- tables, chairs, walls, etc. A point cloud is a way for a program to use its general location (e.g. GPS coordinates or WiFi ID) in conjunction with sensor input such as photographs for localization and to understand the position and structure of objects in the area. This technology is paramount to enable mixed reality computing. Mixed reality (MR) is the merging of real and virtual worlds to produce new environments and visualizations where physical and digital objects co-exist and interact in real time. For this to work, digital objects must understand the context i.e. point cloud of the world it exists in. Mapping the real world is a hard problem. It requires close scanning of the world, up-to-date data, and massive sharing of this data.

Bandwidth Problem

Mobile devices use camera photographs to build a point cloud of their environment. Ideally, these devices could send images to a server to be processed such that the global point cloud would update and the server would send the device universal point cloud data pertaining to its location. However, point clouds are very large. A mobile device presently is unable to load a large area at a time. For moving devices, this requires constant pulling, which means constant searching, through the global point cloud. Sending the volume of images to the central server required to do this in real time is not feasible. The bandwidth required to send this much data coupled with the total latency of sending, processing, and receiving presently makes this not possible. Thus, bandwidth is a major problem.

Mobile Compute Problem

As bandwidth makes sending all images to a server for processing infeasible, a solution is to process the images locally on device, generate small point cloud data to send to the point cloud server, and stream data back. However, mobile compute has limitations in not being able to scale due to device constraints. As mobile devices, such as the iPhone X with six total cores, become more powerful, processing on-device becomes realistic. However, device-specific programming limits the generality of the solution.

Search Problem

Matching local point cloud features to that in the point cloud server is difficult, as the data may be massive and vary somewhat with each other. Optimally searching through oint cloud data for a match must be done efficiently to maintain real time streaming.

Update Problem

Multiple people experiencing in the same area need a shared experience -- that is, much like in a multiplayer videogame, the state must be consistent across devices. This necessitates the ability to massively update the point cloud on each local device when new changes propogate in the global point cloud.

The Challenge

The challenge therefore is to develop a program to efficiently create on a mobile device a point cloud from images, leveraging parallel capabilities of the latest smart phones. Furthermore, we must efficiently search for and match local point cloud features to that in the server, either in the cloud or on device. We must lastly atomic update the point cloud and push updated information in real time.

Point Cloud Creating on Device

Leveraging existing APIs, we will seek to generate point clouds of an environment on iPhones very quickly by using many cores at once. Here is big room for speedup in both the processing of images, and in reduced latency by decreasing the data send/recv with the server.

Scalable Cloud Updates

We will need to implement a thread-safe point cloud data structure, that can scale to accept many pushes and search requests, and keeps mobile devices "state-coherent" by pushing relevant updates.

Resources

We will be starting with Apple's ARKit, which has boilerplate point cloud creation tools, and modify it for our needs. Ideally, this will provide a lot of the framework for our project and reduce overhead, allowing us to focus specifically on speedup. Furthermore, we will be using an iPhone X as it has many avenues of compute for us to explore in speeding up our project.

Goals

We plan to achieve an optimized, fast point cloud generator, update a global point cloud, and update local point clouds in real time. If we are unable to accomplish everything we wish, a reasonable fall-back goal is to design and implement simply the point cloud creation on an iPhone X. This would eliminate all the overhead and increased difficulty of actually pushing/pulling a point cloud from a server and maintaining a thread-safe data structure.

Stretch Goals

In regard to what we hope to achieve, we would like to create an app that anyone can download to experience a shared world. The would include some basic digital interaction between users.

Demo & Presentation

At the poster session, we plan on having a demonstration. If we are able to accomplish our plans, we will demonstrate on two devices that one device capturing point cloud changes propogates on the other device seamlessly. A possible example of this would be placing a virtual object on a table, then turning away from it. A second user would move the table and the ball, and when the first turns back around they would have this updated information.

Platform

<Languages, API, etc -- TBD> "Describe why the platform (computer and/or language) you have chosen is a good one for your needs. Why does it make sense to use this parallel system for the workload you have chosen?"

Schedule

Week of 10/30/2017: Finalize project idea and plan, begin research

Submitted project idea after initial research, and began looking deeply into existing cryptocurrencies.

Week of 11/06/2017: Finish research and complete initial design of proof of work algorithm

Researched heavily, and found little luck in the fact that implementation required much more rigorous cryptography than anticipated. We found that our main idea was implemented already in Monero, a popular cryptocurrency. Thus, we began exploring other related avenues.

Week of 11/13/2017: Create sequential version of algorithm and coin

Update: change project from Anti-parallel Cryptocurrency to Real Time Point Cloud Search, Upload, and Update

We ended up pivoting this week to the project seen above. We got into ARkit and began developing with it. A proof of concept demo has been created using ARKit to detect and define horizontal planes using iPhone. A sufficient ARKit package called ARPointCloud has been determined as the way to move forward. Work has been started on making the demo with ARPointCloud. However, documentation is scarce.

Week of 11/20/2017: Complete Fallback Goal -- Point Cloud on iPhone X, Parallelized

Week of 11/27/2017: Create the server mechanism that scales so multiple users can use the same point cloud

Week of 12/04/2017: Optimize & speedup big time, and create a basic app to demo our work

Week of 12/11/2017: Put finishing touches on the presentation & writeup and present our project

As of right now, because we have recently pivoted from our original project of creating a complete cryptocurrency, we have been doing a lot of research and as such we don't have preliminary results. Regardless, we have a working ARKit demo, and we believe that we are on-track. A very good nice-to-have is the working demo, which we believe is well within the realm of possibility. However, we don't believe that we will be able to deliver a complete cryptocurrency package that is ready for use.

Progress

We researched many cryptocurrencies and their associated proof of work algorithms. We targeted Monero's PoW algorithm specifically, as it is GPU and ASIC resistant by using a large 2MB buffer of memory that is randomly accessed as part of the algorithm. This makes a CPU mine efficiently as 2MB fits into the L3 cache on most CPUs, while GPUs and ASICs cannot support this and must fetch often from memory. After determining a new direction for our project, much research was done into the uses and feasibility of a point-cloud based cryptocurrency.

Richard has created a proof of concept demo with ARKit which involves detecting and illustrating horizontal planes using an ARKit enabled iPhone. He has also done research into APIs which are suitable for point clouds in ARKit, which after research, the best option seems to be ARPointCloud. He attempted to create an analogous demo which used ARPointCloud to detect points of interest in the environment, but is as of yet unable to get it to work. Once this proof of concept and demo is working, then the next step is to have this be offloaded into the cloud, and have many devices be

able to read and write to it at the same time.

Concerning Issues

We are concerned with being able to test the scalability of our project. In an ideal world, we would have able to have thousands of mobile phones updating and getting updates from our central point cloud database. However, we only have 2 iOS devices between the two partners in the project. Other than that, perhaps object permanence is an issue. All demos created so far and most used to demonstrate the capabilities of ARKit start every session from scratch, and do not retain information from previous sessions. We hope that with a central point cloud, we will be able to have object permanence, but we are not yet sure.