Measuring Curves with Augmented Reality

Jacob Young

Overview

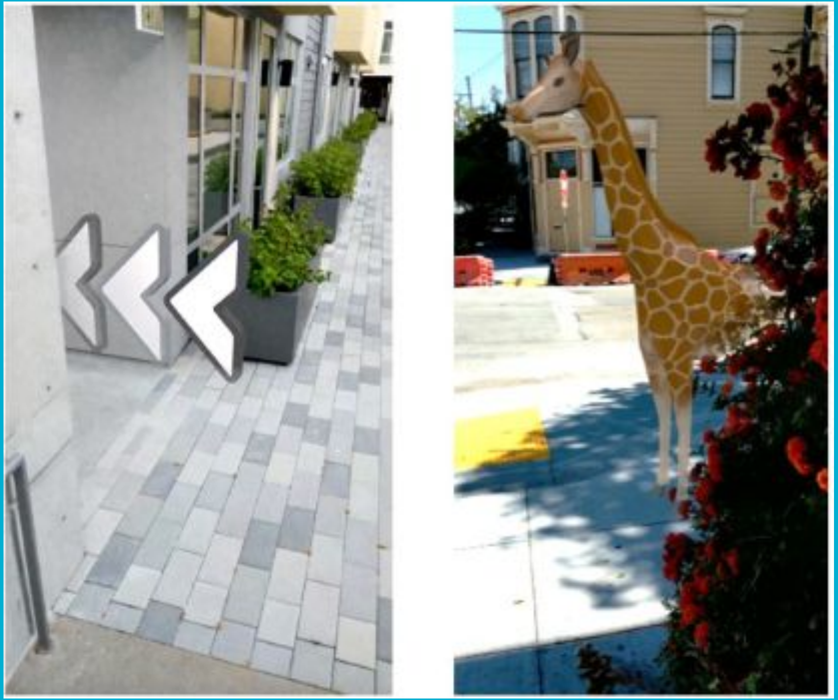
The goal is to measure curves using augmented reality. This is only possible thanks to the authors of the following papers:

1. Depth from Motion
2. Depth Labs
3. Fitting B-Spline Curves to Point Clouds

We’ll take a deeper look into these papers.

Depth from Motion

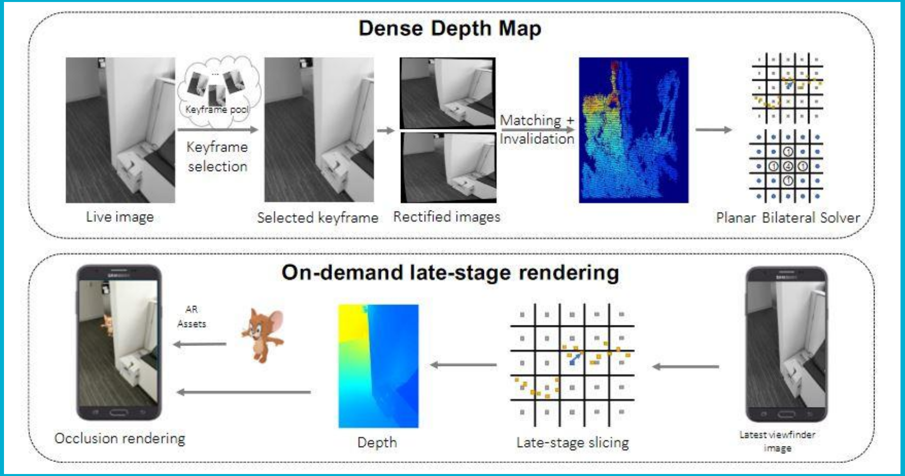
In 2018, Julien Valentin *et al.* introduced a novel method for generating depth maps. This method solves occlusion rendering on older devices that don't have built in depth-sensors. These depth maps, generated per frame, drastically enhance in-app immersion by making virtual objects fit into the real world.



Depth from Motion Process

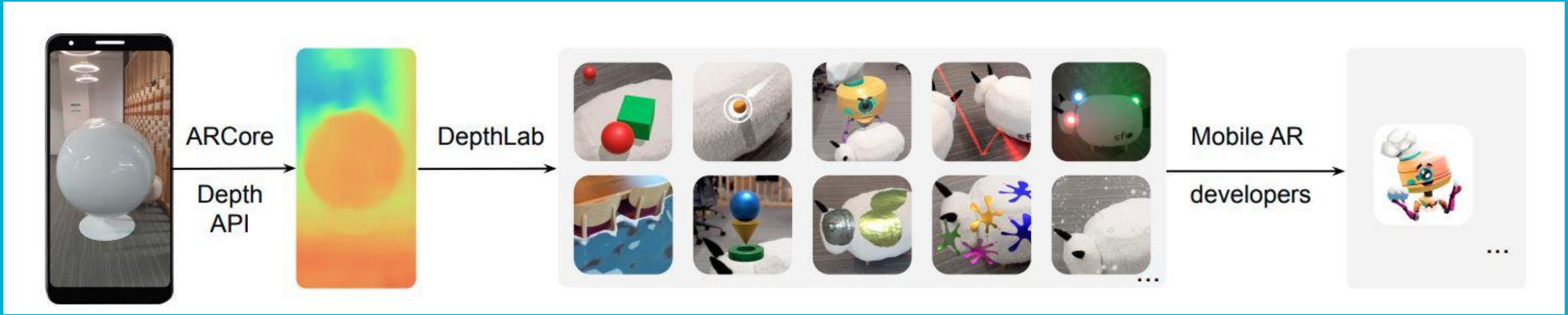
Accomplishing depth from motion can be broken down into 5 steps:

1. Keyframe Selection
2. Matching and Invalidation
3. Planar Bilateral Solving
4. Late-stage slicing
5. Depth map generation



Depth Labs

Introduced by Ruofei Du *et al.* at Google, *Depth Labs* builds directly on top of *Depth from Motion*. The key issue they address is that depth maps generated by Valentin *et al.* require further expertise before developers can use them for key features such as collision detection and occlusion. They have integrated their Depth API into Unity 3D game engine, which is where I have chosen to make my project.



Depth Labs Process.

Depth labs takes output depth data from *Depth from Motion* and organizes it into 3 different categories each with their own data structures and use cases:

1. **Localized Depth** - [Depth Array]

   Operates on a small number of points directly on the **CPU**

   Useful for: Physical measurements, normal vectors estimation and navigation

2. **Surface Depth** - [Depth Mesh]

   Leverages **CPU** or compute shaders on **GPU**

   Useful for: Collision, Physics and Geometry-aware shadows

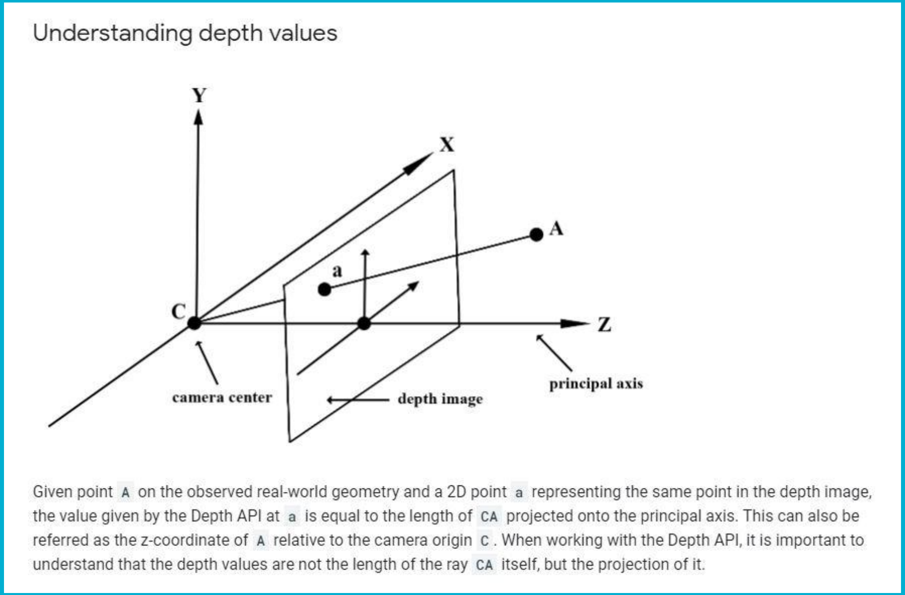
3. **Dense Depth** - [Depth Texture Map]

   Depth data is copied to a **GPU** texture

   Useful for: Screen-space occlusion, re-lighting and photography tasks

Using localized depth maps

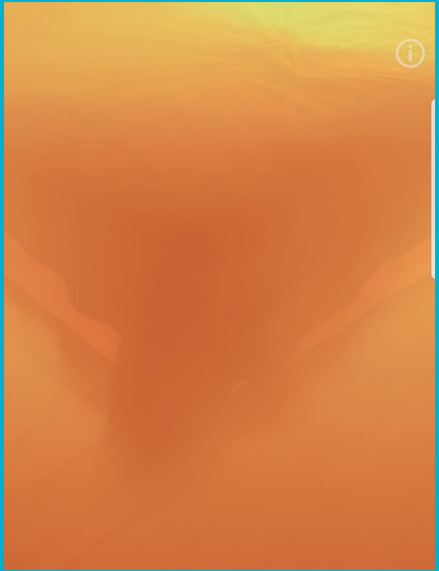
I’ve chosen to use Localized Depth maps, particularly because they are useful for computing physical measurements. Measurements stored in the depth map must be converted from a projected ray **CA** into world space.



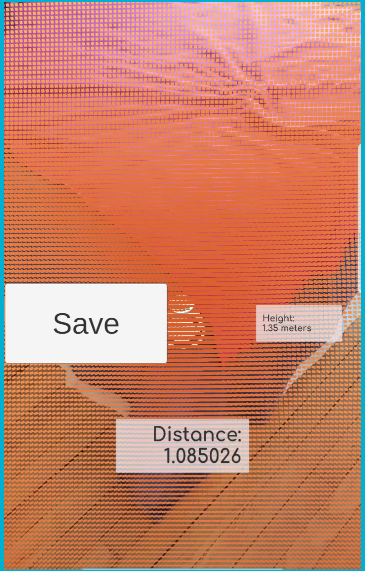
Understanding depth values

Given point `A` on the observed real-world geometry and a 2D point `a` representing the same point in the depth image, the value given by the Depth API at `a` is equal to the length of `CA` projected onto the principal axis. This can also be referred as the z-coordinate of `A` relative to the camera origin `C`. When working with the Depth API, it is important to understand that the depth values are not the length of the ray `CA` itself, but the projection of it.

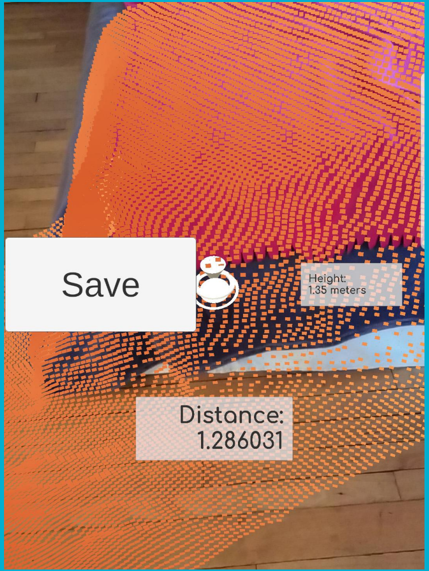
Creating a point cloud.



Depth map read in.

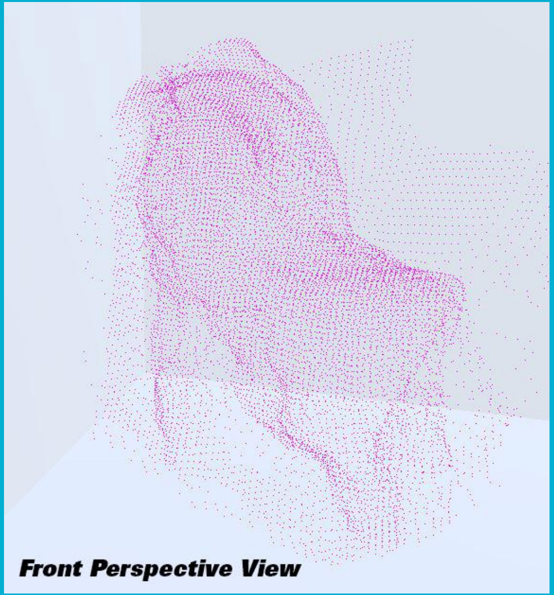


Depth values screen-space projected into world space.



Another perspective of world space point cloud.

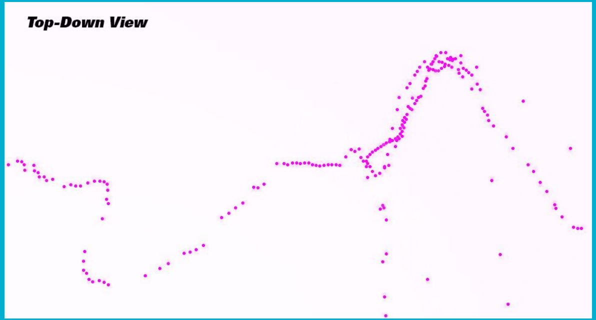
Measuring perimeter.



Side Perspective View

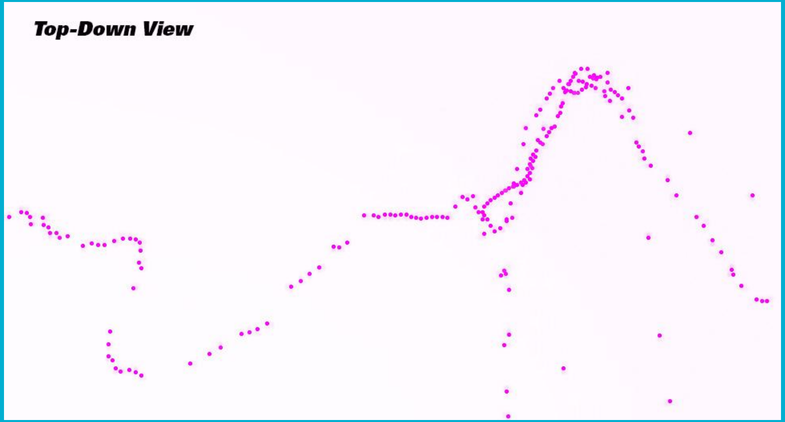
Point cloud of a scanned chair.

We take a horizontal slice of our scanned object from an arbitrary height to simply approximating the perimeter in 2 dimensions.

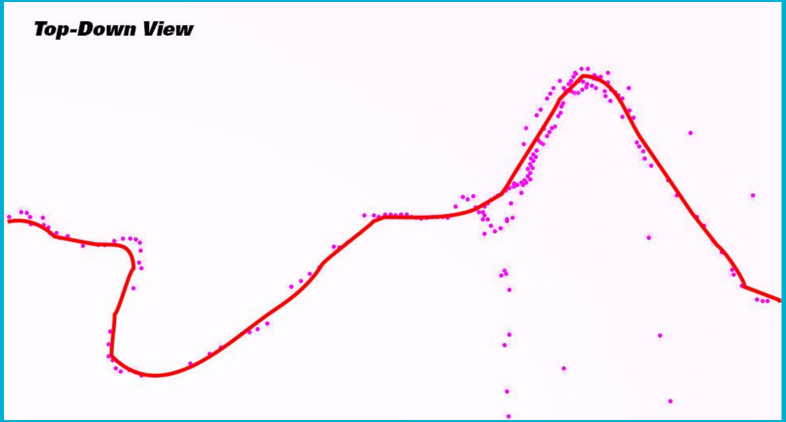


Measuring perimeter.

Ideally, we want to go from this raw point cloud to a smooth approximation of the arc length, but how?



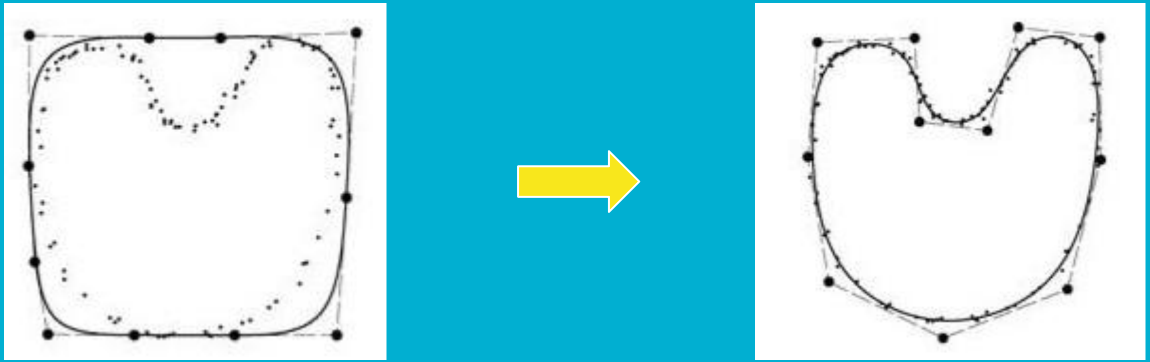
Horizontal slice of chair point cloud.



Arc length Approximation.

B-Spline fitting to Point Clouds

In order to solve this arc length approximation problem, we must treat it as a least-squares optimization problem. Wenping Wang et al. present an efficient method for approximating a target shape defined by an unorganized point cloud called Squared Distance Minimization (SDM).



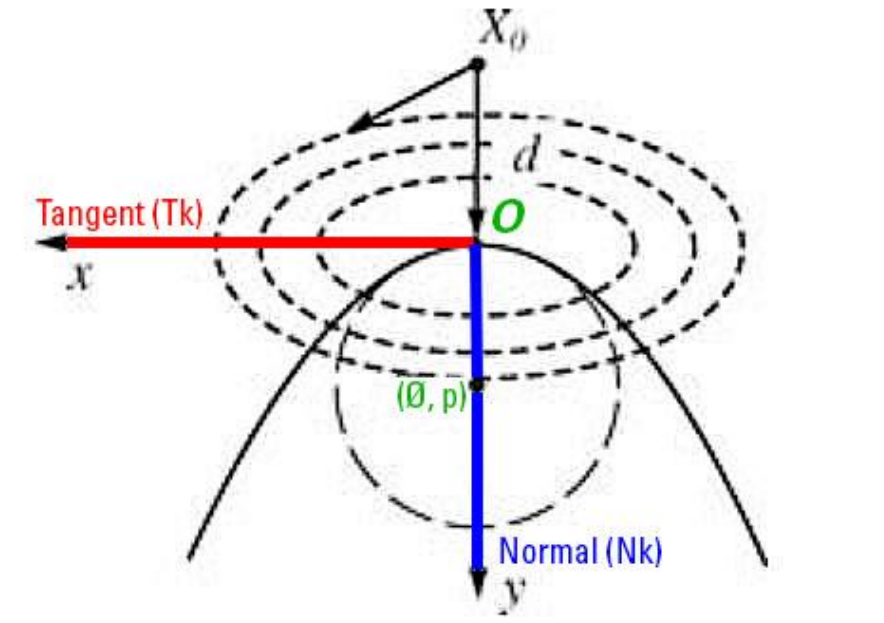
Squared Distance Minimization

SDM fundamentally works by taking the frenet frame (the tangent and normal vector) and computing a squared distance error **eSDk(D)** term which iteratively adjusts the initial curve to approximate the point cloud shape.

X0 is a given point in a point cloud

O is the control point of the curve which is closest to *X0*.

p is the radius of curvature



$$e_{SD,k}(\mathcal{D}) = \begin{cases} \frac{d}{d-\rho}[(P_+(t_k) - X_k)^T T_k]^2 + [(P_+(t_k) - X_k)^T N_k]^2, & \text{if } d < 0, \\ [(P_+(t_k) - X_k)^T N_k]^2, & \text{if } 0 \le d < \rho, \end{cases}$$

SDM Process

Wang et al. claim that SDM is a two step process starting with:

1. Specify a proper initial shape of a B-Spline fitting curve
2. Compute SD error terms for all data points.

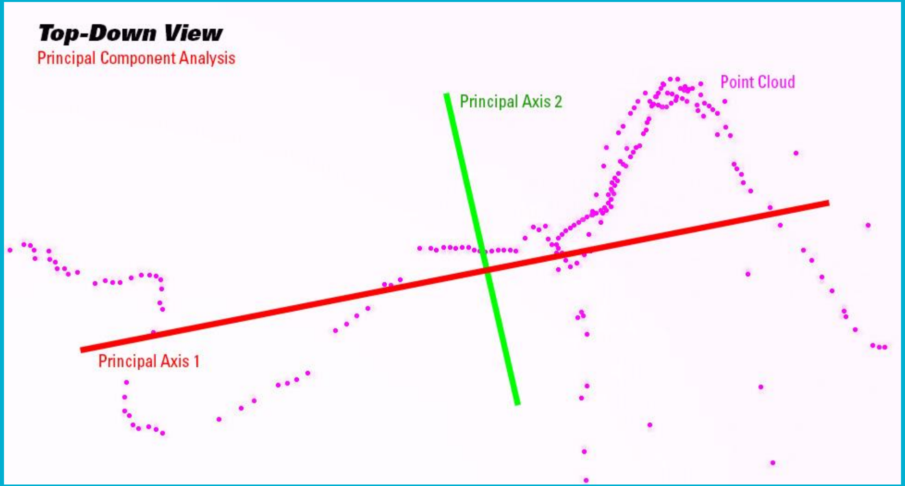
But they do not mention the constraints defining a proper initial shape nor how to achieve it. Luckily, Abhishek Pandey, author of *3D Surface Approximation from Point Clouds* finds that by using Principal Component Analysis decomposition we can define a proper initial curve from the dominant axis of the point cloud.

A properly initialized curve

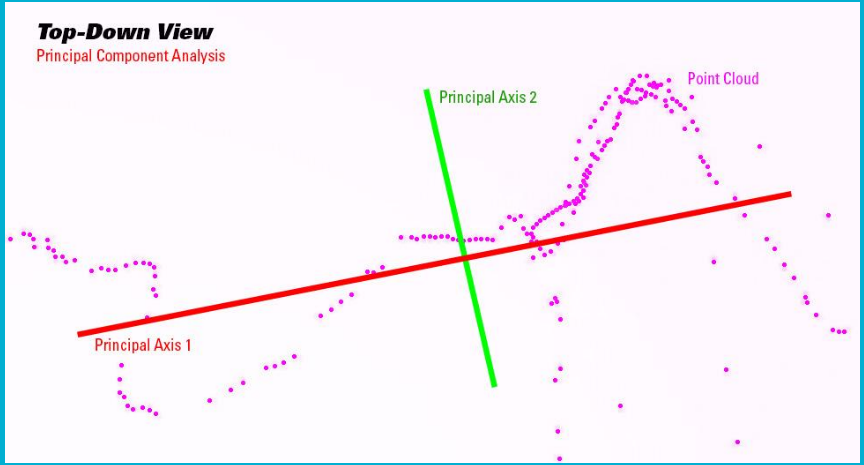
Using Principal Component Analysis (PCA), we identify the dominant axes of the point cloud data slice.

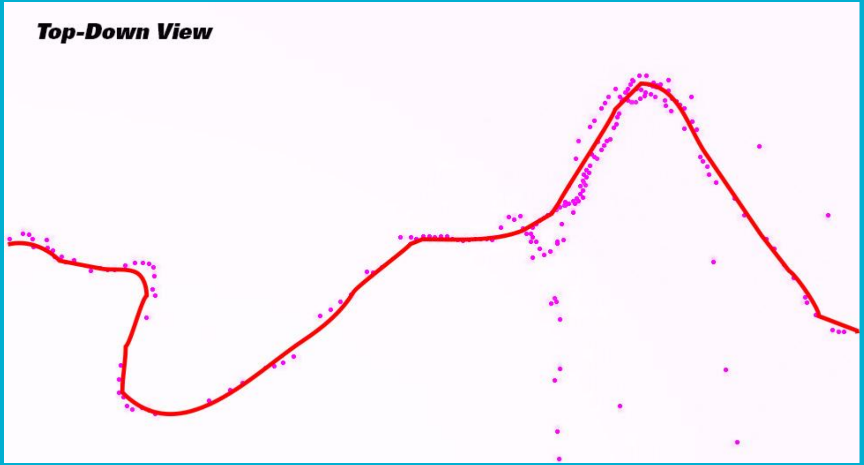
This is done by performing Eigen-Decomposition on the Covariance Matrix of the data to retrieve the Eigenvectors (the principal axes).

The main principal axis is used as the initial curve.



Remaining Work:





SDM still needs to be implemented in the project to determine the accuracy of the arc length approximation.

Thank you

References

1. Julien Valentin, Adarsh Kowdle, Jonathan T. Barron, Neal Wadhwa, Max Dzitsiuk, Michael, Schoenberg, Vivek Verma, Ambrus Csaszar, Eric Turner, Ivan Dryanovski, Joao Afonso, Jose Pascoal, Konstantine Tsotsos, Mira Leung, Mirko Schmidt, Onur Guleryuz, Sameh Khamis, Vladimir Tankovitch, Sean Fanello, Shahram Izadi, and Christoph Rhemann. Depth from motion for smartphone ar. 2018.

2. Ruofei Du, Eric Turner, Maksym Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Dourgarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Cruces, Shahram Izadi, Adarsh Kowdle, Konstantine Tsotsos, and David Kim. DepthLab: Real-time 3D Interaction with Depth Maps for Mobile Augmented Reality. In Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology, UIST. ACM, 2020.

3. Helmut Pottmann Wenping Wang and Yang Liu. Fitting b-spline curves to point clouds by curvature-based squared distance minimization. ACM Transactions on Graphics, 25(2):214–238, 2006.

4. Abhishek Pandey. 3d surface approximation from point clouds. 2019.