DepthLab: Real-time 3D Interaction with Depth Maps for Mobile Augmented Reality

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, David Kim

Google | ACM UIST 2020
Introduction
Mobile Augmented Reality
Introduction

Google’s ARCore
Introduction

Google’s ARCore
Is direct placement and rendering of 3D objects sufficient for realistic AR experiences?
Not always!
Virtual content looks like it’s “pasted on the screen” rather than “in the world”!
Introduction

Motivation
Introduction
How can we bring these advanced features to mobile AR experiences without relying on dedicated sensors or the need for computationally expensive surface reconstruction?
Google
• Pixel 2, Pixel 2 XL, Pixel 3, Pixel 3 XL, Pixel 3a, Pixel 3a XL, Pixel 4, Pixel 4 XL

Huawei
• Honor 10, Honor V20, Mate 20 Lite, Mate 20, Mate 20 X, Nova 3, Nova 4, P20, P30, P30 Pro

LG
• G8X ThinQ, V35 ThinQ, V50S ThinQ, V60 ThinQ 5G

OnePlus
• OnePlus 6, OnePlus 6T, OnePlus 7, OnePlus 7 Pro, OnePlus 7 Pro 5G, OnePlus 7T, OnePlus 7T Pro

Oppo
• Reno Ace

Samsung
• Galaxy A80, Galaxy Note8, Galaxy Note9, Galaxy Note10, Galaxy Note10 5G, Galaxy Note10+, Galaxy Note10+ 5G, Galaxy S8, Galaxy S8+, Galaxy S9, Galaxy S9+, Galaxy S10e, Galaxy S10, Galaxy S10+, Galaxy S10 5G, Galaxy S20, Galaxy S20+ 5G, Galaxy S20 Ultra 5G

Sony
• Xperia XZ2, Xperia XZ2 Compact, Xperia XZ2 Premium, Xperia XZ3

Xiaomi
• Pocophone F1

And growing…

https://developers.google.com/ar/discover/supported-devices
Is there *more* to realism than occlusion?
Surface interaction?
Realistic Physics?
Introduction

Path Planning?
DepthLab: Real-time 3D Interaction with Depth Maps for Mobile Augmented Reality

Ruofei Du, Eric Turner, Max Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Dourgarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Cruces, Shahram Izadi, Adarsh Kowdle, Konstantine Tsotsos, David Kim

Google | ACM UIST 2020
Introduction
Related Work
Valentin et al.

Live image → Keyframe pool → Keyframe selection → Selected keyframe → Matching + Invalidation → Stereo Matching → Filtering and smoothing → Depth → Call to Depth API → Occlusion rendering → Asset from application → Depth from Motion for Smartphone AR (SIGGRAPH Asia 2018)
Introduction

Depth Lab
Introduction
Introduction
Related Work
Valentin et al.

Live image → Selected keyframe → Stereo Matching → Depth → Occlusion rendering

Keyframe pool → Keyframe selection → Matching + Invalidation → Filtering and smoothing

Asset from application

Depth from Motion for Smartphone AR (SIGGRAPH Asia 2018)
Introduction

Depth Lab

Up to 8 meters, with the best within 0.5m to 5m
Motivation

Gap from raw depth to applications
Design Process
3 Brainstorming Sessions

3 brainstorming sessions
18 participants
39 aggregated ideas
Supplementary Material for DepthLab: Real-time 3D Interaction with Depth Maps for Mobile Augmented Reality

3 Brainstorming Sessions
System
Architecture overview

Input
- Camera Image
- Phone Orientation
- Camera Parameters
  - focal length
  - intrinsic matrix
  - extrinsic matrix
  - projection matrix

DepthLab Data Structures and Utilities
- Conversion Utilities
  - screen uv/xy → depth
  - screen uv/xy ↔ world vertex
  - screen uv/xy → local normal
  - screen uv/xy → world normal
  - depth uv ↔ depth xy
  - screen uv ↔ screen xy

Depth Array

Depth Map
- Depth Mesh
- Depth Texture

DepthLab Algorithms
- Localized Depth
  - orientation
  - hit test
  - reflection
- Surface Depth
  - physics
  - texture decal
  - depth mesh
- Dense Depth
  - relighting
  - occlusion
  - aperture
Data Structure

Depth Array

2D array (160x120 and above) of 16-bit integers
Data Structure

Depth Mesh

(a) input depth map
(b) template mesh
(c) real-time depth mesh

winding order of the template mesh
Data Structure

Depth Texture
<table>
<thead>
<tr>
<th></th>
<th>Localized Depth</th>
<th>Surface Depth</th>
<th>Dense Depth</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
<td>✅</td>
<td>✅</td>
<td>✗ (non-real-time)</td>
</tr>
<tr>
<td><strong>GPU</strong></td>
<td>N/A</td>
<td>☑ (compute shader)</td>
<td>☑ (fragment shader)</td>
</tr>
<tr>
<td><strong>Prerequisite</strong></td>
<td>point projection normal estimation</td>
<td>depth mesh triplanar mapping</td>
<td>anti-aliasing multi-pass rendering</td>
</tr>
<tr>
<td><strong>Data Structure</strong></td>
<td>depth array</td>
<td>depth mesh</td>
<td>depth texture</td>
</tr>
</tbody>
</table>
| **Example Use Cases** | physical measure | collision & physics | scene relighting
|                  | oriented 3D cursor | virtual shadows | aperture effects
|                  | path planning      | texture decals | occluded objects |
Localized Depth
Coordinate System Conversion

Conversion Utilities

screen uv/xy → depth
screen uv/xy ↔ world vertex
screen uv/xy → local normal
screen uv/xy → world normal
depth uv ↔ depth xy
screen uv ↔ screen xy
Localized Depth
Normal Estimation

\[ n_p = (v_p - v_{p+(1,0)}) \times (v_p - v_{p+(0,1)}) \]
Algorithm 1: Estimation of the Normal Vector of a Screen Point in DepthLab.

Input : A screen point \( p \leftarrow (x, y) \) and focal length \( f \).
Output : The estimated normal vector \( \mathbf{n} \).

1. Set the sample radius: \( r \leftarrow 2 \) pixels.
2. Initialize the counts along two axes: \( c_x \leftarrow 0, c_y \leftarrow 0 \).
3. Initialize the correlation along two axes: \( \rho_x \leftarrow 0, \rho_y \leftarrow 0 \).
4. for \( \Delta x \in [-r, r] \) do
   for \( \Delta y \in [-r, r] \) do
     Continue if \( \Delta x = 0 \) and \( \Delta y = 0 \).
     Set neighbor's coordinates: \( q \leftarrow [x + \Delta x, y + \Delta y] \).
     Set \( q \)'s distance in depth: \( d_{pq} \leftarrow \|D(p), D(q)\| \).
     Continue if \( d_{pq} = 0 \).
     if \( \Delta x \neq 0 \) then
       \( c_x \leftarrow c_x + 1 \).
       \( \rho_x \leftarrow \rho_x + d_{pq} / \Delta x \).
     end
     if \( \Delta y \neq 0 \) then
       \( c_y \leftarrow c_y + 1 \).
       \( \rho_y \leftarrow \rho_y + d_{pq} / \Delta y \).
     end
   end
20. Set pixel size: \( \lambda \leftarrow \frac{D(p)}{f} \).
21. return the normal vector \( \mathbf{n} : \left(-\frac{\rho_y}{\lambda c_y}, -\frac{\rho_x}{\lambda c_x}, -1\right) \).
Localized Depth

Normal Estimation
Surface Depth

Use Cases

(a) input depth map
(b) template mesh
(c) real-time depth mesh

winding order of the template mesh
Physics with depth mesh.
Texture decals with depth mesh.
Projection mapping with depth mesh.
Dense Depth

Depth Texture - Antialiasing
Dense Depth
Real-time relighting

\[ L \quad N \quad \theta \]
Dense Depth
Why normal map does not work?

Input: Depth map \( D \), the camera image \( I \), camera intrinsic matrix \( K \), \( L \) light sources \( L = \{ L_i | i \in L \} \) with each light’s location \( v_L \) and intensity in RGB channels \( \phi_L \).

Output: Relighted image \( O \).

for each image pixel \( p \) in depth map \( D \) in parallel do

1. Sample \( p \)'s depth value \( d \leftarrow D(p) \).
2. Compute the corresponding 3D vertex \( v_p \) of the screen point \( p \) using the camera intrinsic matrix \( v_p \) with \( K \):
   \[ v_p = D(p) \cdot K^{-1} \cdot [p, 1]^T \]
3. Initialize relighting coefficients of \( v_p \) in RGB: \( \phi_p \leftarrow 0 \).

for each light \( L_i \in \text{light sources} \) do

1. Set the current photon coordinates \( v_o \leftarrow v_p \).
2. Set the current photon energy \( E_o \leftarrow 1 \).

while \( v_o \neq v_L \) do

1. Compute the weighted distance between the photon to the physical environment
   \[ \Delta d \leftarrow \alpha |v_o^* - v_L^*| + (1 - \alpha) |v_o - v_p|, \alpha = 0.5. \]
2. Decay the photon energy: \( E_o \leftarrow 95\% E_o \)
3. Accumulate the relighting coefficients:
   \[ \phi_p \leftarrow \phi_p + \Delta E_o \phi_L \]
4. March the photon towards the light source:
   \[ v_o \leftarrow v_o + (v_L - v_o)/S, \text{here } S = 10, \text{ depending on the mobile computing budget.} \]

end

Sample pixel’s original color: \( \Phi_p \leftarrow I(p) \).

Apply relighting effect:

\[ O(p) \leftarrow \gamma |0.5 - \phi_p| \cdot \Phi_p^{1.5 - \phi_p} - \Phi_p, \text{ here } \gamma \leftarrow 3. \]
Dense Depth
Wide-aperture effect
Dense Depth
Occlusion-based rendering
Experiments

DepthLab minimum viable application
<table>
<thead>
<tr>
<th>Procedure</th>
<th>Timings (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DepthLab's overall processing and rendering in Unity</td>
<td>8.32</td>
</tr>
<tr>
<td>DepthLab's data structure update and GPU uploading</td>
<td>1.63</td>
</tr>
<tr>
<td>Point Depth: normal estimation algorithm</td>
<td>&lt; 0.01</td>
</tr>
<tr>
<td>Surface Depth: depth mesh update algorithm</td>
<td>2.41</td>
</tr>
<tr>
<td>Per-pixel Depth: visualization with single texture fetch</td>
<td>0.32</td>
</tr>
</tbody>
</table>
Experiments

Relighting

- Input color vs. output with #samples=8
- Input depth vs. output with #samples=128

Graph showing rendering time per frame vs. number of samples per ray:
- Rendering time increases significantly with the number of samples per ray.
Experiments
Aperture effects

(a) examples of aperture effects
(b) performance benchmark
Discussion
Deployment with partners
Discussion
Deployment with partners
Discussion
Deployment with partners
Limitations

*Design space of dynamic depth*

Dynamic Depth? HoloDesk, HyperDepth, Digits, Holoportation for mobile AR?
Envision
Design space of dynamic depth
ARCore Depth Lab - Depth API Samples for Unity

Copyright 2020 Google LLC. All rights reserved.

ARCore Depth Lab is a set of ARCore Depth API samples that provides assets using depth for advanced geometry-aware features in AR interaction and rendering. Some of these features have been used in this Depth API overview video.

ARCore Depth API is enabled on a subset of ARCore-certified Android devices. iOS devices (iPhone, iPad) are not supported. Find the list of devices with Depth API support (marked with Supports Depth API) here: https://developers.google.com/ar/discover/supported-devices. See the ARCore developer documentation for more information.

Download the pre-built ARCoreDepth Lab app on Google Play Store today.

Sample features

The sample scenes demonstrate three different ways to access depth:

1. Localized depth: Sample single depth values at certain texture coordinates (CPU).
   - Character locomotion on uneven terrain
   - Collision checking for AR object placement
   - Laser beam reflections
   - Oriented 3D reticles
DepthLab: Real-time 3D Interaction with Depth Maps for Mobile Augmented Reality

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, David Kim

Google | ACM UIST 2020
DEPTHLAB: REAL-TIME 3D INTERACTION WITH DEPTH MAPS FOR MOBILE AUGMENTED REALITY

ACM UIST 2020

Download PDF (6.4 MB) or Low-Res PDF (6 MB)

Ruofei Du, Eric Turner, Maksym Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Pourgarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Cruces, Shahram Izadi, Adarsh Kowdle, Konstantine Tsotsos, David Kim

Google LLC
Introduction

Depth Map
Introduction

(a) oriented reticles and splats
(b) ray-marching-based scene relighting
(c) depth visualization and particles
(d) geometry-aware collisions
(e) 3D-anchored focus and aperture effect
(f) occlusion and path planning
Thank you!

www.duruofei.com

DepthLab: Real-Time 3D Interaction With Depth Maps for Mobile Augmented Reality

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


pdf, lowres | website, code, demo, supp | cite
Occlusion is a critical component for AR realism!
Correct occlusion helps ground content in reality, and makes virtual objects feel as if they are actually in your space.
Algorithm 2: Real-time Depth Mesh Generation.

**Input**: Depth map $D$, its dimension $w \times h$, and depth discontinuity threshold $\Delta d_{\text{max}} = 0.5$.

**Output**: Lists of mesh vertices $\mathbb{V}$ and indices $\mathbb{I}$.

*In the initialization stage on the CPU:*

1. **for** $x \in [0, w - 1]$ **do**
2.   **for** $y \in [0, h - 1]$ **do**
3.     Set the pivot index: $I_0 \leftarrow y \cdot w + x$.
4.     Set the neighboring indices:
5.        $I_1 \leftarrow I_0 + 1$, $I_2 \leftarrow I_0 + w$, $I_3 \leftarrow I_2 + 1$.
6.     Add the vertex $(x/w, y/h, 0)$ to $\mathbb{V}$.
7. **end**

*In the rendering stage on the CPU or GPU:*

8. **for each vertex $v \in \mathbb{V}$ do**
9.    Look up $v$’s corresponding screen point $p$.
10. Fetch $v$’s depth value $d_0 \leftarrow D(p)$.
11. Fetch $v$’s neighborhoods’ depth values:
12.     $d_1 \leftarrow D(p + (1, 0))$, $d_2 \leftarrow D(p + (0, 1))$,
13.     $d_3 \leftarrow D(p + (1, 1))$.
14. Compute average depth $\bar{d} \leftarrow \frac{d_0 + d_1 + d_2 + d_3}{4}$.
15. Let $d \leftarrow [d_0, d_1, d_2, d_3]$.
16. **if any** $(\text{step}(\Delta d_{\text{max}}, |d - \bar{d}|)) = 1$ **then**
17.     Discard $v$ due to large depth discontinuity.
18. **end**
Localized Depth

Avatar Path Planning
Dense Depth

Depth Texture

(a) relighting effect  (b) aperture effect  (c) fog effect
Introduction

Depth Map

(a) input depth map (bilinearly filtered)
(b) result depth map with FXAA
(c) result with depth-guided FXAA
<table>
<thead>
<tr>
<th>Taxonomy</th>
<th>Depth Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Depth-based Interaction Design Space</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Geometry-aware</strong></td>
<td><strong>Depth Interaction</strong></td>
</tr>
<tr>
<td>Rendering</td>
<td>Interface</td>
</tr>
<tr>
<td>occlusion</td>
<td>3D cursor</td>
</tr>
<tr>
<td>shadows</td>
<td>bounding-box</td>
</tr>
<tr>
<td>relighting</td>
<td>region selection</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td><strong>Actions</strong></td>
<td><strong>Gestures</strong></td>
</tr>
<tr>
<td>physics</td>
<td>static hand</td>
</tr>
<tr>
<td>path planning</td>
<td>dynamic motion</td>
</tr>
<tr>
<td>free-space check</td>
<td>3D touch</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Figure 12. Given a dense depth texture, a camera image, and virtual light sources, we altered the lighting of the physical environment by tracing occlusions along the light rays in real time.
Figure 13. Wide-aperture effect focused on a world-anchored point on a flower from different perspectives. Unlike traditional photography software, which only anchors the focal plane to a screen point, DepthLab allows users to anchor the focal point to a physical object and keep the object in focus from even when the viewpoint changes. Please zoom in to compare the focus and out-of-focus regions.