Web AR 各类方案探讨

作者：艾孜尔江·艾尔斯兰
时间：2020 年 7 月 16 日星期四

一、 AR.js

使用 AR.js 过程中，发现其主要通过识别卡片进行 AR 效果的呈现。

它是基于 ARToolKit 来实现诸如标记图检测和平面检测那样对于 AR 体验来说至关重要的视觉系统的。AR.js 支持任何带有摄像头的设备，例如移动设备和桌面设备。AR.js 能够在桌面和移动设备上以 60 帧/秒的速度提供 AR 体验。在与 3D 模型交互时，无论是在桌面还是移动设备上，性能都不会下降，这很可能是由于它是基于标记图的系统（Marker-based System）来识别平面的，也正因如此，AR.js 可以实现快速高效的平面检测和模型定位。它还能让参与者在不控制使用设备的情况下与应用程序进行互动，例如，教室里的孩子们在桌子上移动标记，投射的内容对互动也会产生反应。然而，这种基于标记图的系统的一个很大痛点是，标记图必须始终清晰地在视野范围内，以便模型能够正确地显示。

二、 A-Frame AR

A-Frame 使用简单的基于 html 的实体-组件结构，使应用程序开发快速方便。然而，A-Frame AR 只能在那些实验性浏览器，比如谷歌的 Web AR on AR Kit/Core 和 Mozilla 的 WebXR Viewer 上使用。A-Frame AR 只支持上述实验性浏览器。用户可以在 Web AR on AR Kit 和 Web XR Viewer 上运行基于 A-Frame AR 的各类应用程序，在这两个浏览器上的体验都是类似的。虽然 A-Frame AR 比 AR.js 具有更多的基于环境的体验，但在性能方面却逊色一些。3D 模型的显示常常滞后，平面检测也有很多缺陷。此外，当添加与模型的交互时，事件和结果之间的延迟较为显著。这可能是由于 AR 浏览器的自身的实验性质造成的，因为在官方的各类试验中，这些浏览器经常因为内存不足而崩溃。A-Frame AR 不是基于标记图的。在某些方面，这允许 A-Frame AR 创造“真实”的 AR 体验，即不完全依赖外部标记。在未来，很多对该技术持有乐观态度的人士认为这将会为其在 Web AR 开发带来很大的灵活性，因为研究人员的主要精力将会花在如何把更高效的计算机视

觉算法合并到库中。然而就目前来说，平面检测缺陷很多，并且 A-Frame AR 的渲染更新效率很低，技术还没有完全到位。

笔者在测试 A-Frame AR 的库时也发现,A-Frame AR 的代码更像是 HTML 标签，起结构大致为，首先在整个 HTML 身体标签之下有一个场景标签<a-scene></ascene>，再场景标签中需要定义场景，也就是屏幕的长和宽，以及影像来源、影像是否内嵌等信息。在这之后就需要定义一个 marker 标签用以设置具体的卡片，因为卡片算是用户在真实场景中呈现虚拟模型的重要支撑和保障，接下去的所有模型的操作都通过卡片标签下的“实体标签”的形式撰写，包括模型实体的具体大小、旋转以及缩放等相关信息。笔者在实验中发现其中可以直接加载的模型后缀是.gltf，也就是说，并非传统的 obj 格式或者说是 fbx 格式。据全球知名图形学组织 The Khronos Group 对于该格式的相关资料记载，glTF 格式本质上是一个 JSON 文件。这一文件描述了整个 3D 场景的内容。它包含了对场景结构进行描述的场景图。场景中的 3D 对象通过场景结点引用网格进行定义。材质定义了 3D 对象的外观，动画定义了 3D 对象的变换操作(比如选择、平移操作)。蒙皮定义了 3D 对象如何进行骨骼变换，相机定义了渲染程序的视锥体设置。在笔者尝试更换里面的模型以检测起能否加载其他类型的模型时发现在终端中会提示加载的模型文件中第一个位置错误——Unexpected token at position 1: # ； 这也就意味着它在加载的时候是需要读取内部的编码进行的，显然，将 obj 文件直接拖进去、把代码中所需要加载的模型名字更换一下或者是粗暴地将模型文件的后缀改成 gltf 格式并不能绕过它加载模型过程中对于内部数据的读取。

AR.js 和 A-Frame AR 比较

Comparison	**AR.js**	**A-Frame AR**
Supporting Libraries	A-Frame, three.js, ARToolKit	A-Frame, three.ar.js, WebARonARKit/Core, WebXR
Mobile Support?	Yes	Yes
Desktop Support?	Yes	No
Supported Mobile Browsers	iOS： Safari Android：Chrome	iOS ：WebARonARKit 、WebXR Viewer Android：WebARonARCore
Latency	Low (< 10 ms for display and interaction)	Medium (low for display, high for interaction)
Marker-based?	Yes	No
Time for Hello World	< 10 minutes	20-30 minutes (including setting up one of the experimental browsers)
Use Cases	1. Informative	1. Object recognition

	marker-based displays 2. Games with marker-based characters and interactions 3. Educational content that can be manipulated by outside participants in real-time	applications 2. 3D AR drawing/manipulation applications 3. Standard A-Frame use cases in AR

三、 JSARToolkit

总体效果上与上面三个类似，也是基于卡片才能实现 AR 效果，因为在最终效果上有着与 AR.js 和 AFrame AR 过多的雷同之处，无论在演示链接上抑或是在本次讨论中笔者不再赘述。

四、 Awe.js

使用 Awe.js 过程中，该库称其为开发者提供了一系列比较好用的接口，但是源码并未从任何渠道找到。并且在使用其可视化编辑器时也并不像前 XR+那样方便。

五、 8th-Wall

核心部分闭源且无法轻易获取，因为需要用其官方给的 app key 作为 url 参数去访问指

定的网址来获取改网址中的 js 源码，然而即便已经拥有 app key，也需要与开发者绑定设备，否则会出现“Device Not Authorized to View ”的情况，这样的防盗技术对破解他们核心源码来说寸步难行，最令人遗憾的是该项目代码的开源部分已经利用前端代码混淆技术进行防盗处理。好在从项目中可以见到其利用的是 A-Frame AR 这个方案，因为在其 HTML 文件中可以发现 A-Frame AR 相关“场景”、“相机”、“实体”等标签，在部分交互相关的代码中也能够看到对 AFRAME 属性的调用。

六、 Wikitude

该库提供了大量看似非常有用的示例，但是无一运行成功，原因在于其核心代码都在官网上，然而关于具体代码的存放位置已经改变，官网没有给出改变的去向。从官网中的栏位标题中可以看出，该方案也是收费的。从它的宣传视频上可以看到它也位用户提供了在线可视化编辑器。

七、 WebXR

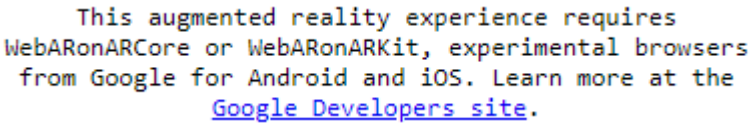
WebXR 是谷歌强推并且在 MDN 上也早有接口的混合现实方案。笔者在其论坛中、技术网站中甚至是宣传网站中都看到其能够较好地识别地面并在任意地面上产生各类模型。但是在下载其源码之后发现并没有任何关于 AR 相关的案例，即便有，也无法以 AR 的形式打开。在这一问题的催促之下本人赶赴其官网中寻找答案，令人百思不得其解的是，官网对 AR 相关的案例给出的这样一个非常明确的解释——“由于目前尚属于内测状态，所有的 AR 功能都没有启用。” 面对这样的说法，笔者顿时感到被耍。在下载下来的源码中确实没有找到任何关于 AR 的 HTML 页面或者案例，好多相关案例要么没有对应的文件——在点击打开页面之后会显示没有相关页面（下载好的仓库中没有， 先前还以为是下载错了，但后来发现官网给出的也是一样的情况）、要么就是在一个莫大的白色屏幕中央显示一行无关痛痒语句“点击‘进入 XR 场景’以进入 XR 场景”；相信读者在本人给出的案例中在观看 WebXR 官方案例的时候定会看到有些案例类似的提示。这种提示令人啼笑皆非的是，并没有任何地方显示可以点击的按钮或类似于按钮的东西，左上角唯一一个能够点击产生交互的按钮被官方

给禁用，令人着实费解。甚至在源码内部也有关于“本源码只适用于实验，对于任何生产需求我们不作保证，在运行时出现一些库和包文件没有得以加载纯属正常现象，望知悉。” 这样的告示因此，且不提 WebXR 在 AR 中能否为我们提供较好的任意平面识别系统，单就从其提供 AR 演示这一点上就留有许多令人质疑之点。当然，作为开发者，面对 WebXR 这样令人无奈的情形，我们只默默地希望谷歌在日后的发展中能够尽快开启 AR 模块，好让我们开发者们能够见到其真容。笔者在文末给出了众多 WebXR 相关资料链接，读者可以作为拓展性阅读去看看其目前的发展生态。

由于其实验性质，网络上 WebXR 资源虽多，笔者仍旧不对其在演示的链接中保留过多。

八、 Three.ar.js

在尝试 three.ar.js 这个库时，发现其所有源码在目前通用的浏览器是无法正常打开运行的，笔者得知，three.ar.js 是基于谷歌自家的 WebARonARKit（ios）和 WebARonARCore（android）。在通用的浏览器上实施将会有如下图所示的提示。



[谷歌官方给出的提示]

九、 xr.plus

提供了可视化编辑面板，让用户可以直接在上面创作，并提供无卡识别功能，操作简便，适合用户使用，未开放源码。而且笔者再该平台上编辑场景并在手机上运行后发现它所谓的“markerless”实际上是假的，初始状态下的模型会生成在平面上，但是当人走过去的时候会出现模型跟着人一起走的情况，也就是模型不会固定在之前识别过的哪个平面上。

十、 vectary

与 XR+类似，拥有极简的编辑界面，但是费用昂贵，未开放源码。

十一、 kivicube

功能比上面那个少一些但是整体上和 vectaryy 的差不多。任意地面识别的功能尚未具备。在官网上对于“平面检测与跟踪”这一块写着“即将到来”等字样。

十二、 beautyar

用在美妆领域较多，使用的是面部识别算法检测人脸。

十三、 zap.works

zap.works 提供了可以用于 Threejs、Unity 以及 AFrame AR 的 SDK 供开发者使用。源码地址： https://github.com/zappar-xr 。

其源码中有点击地面时固定模型于地面中的方式，但是在真实环境下测试时无论是手机抑或是电脑端，都没有任何效果。卡片识别的案例则是可以正常运行的，跟 AR.js、AFrame AR 的效果一致。npm 中还有它的 cli 工具，利用它的工具还能直接通过 zapworks train myImage.png 命令来生成卡片。

十四、 model-viewer

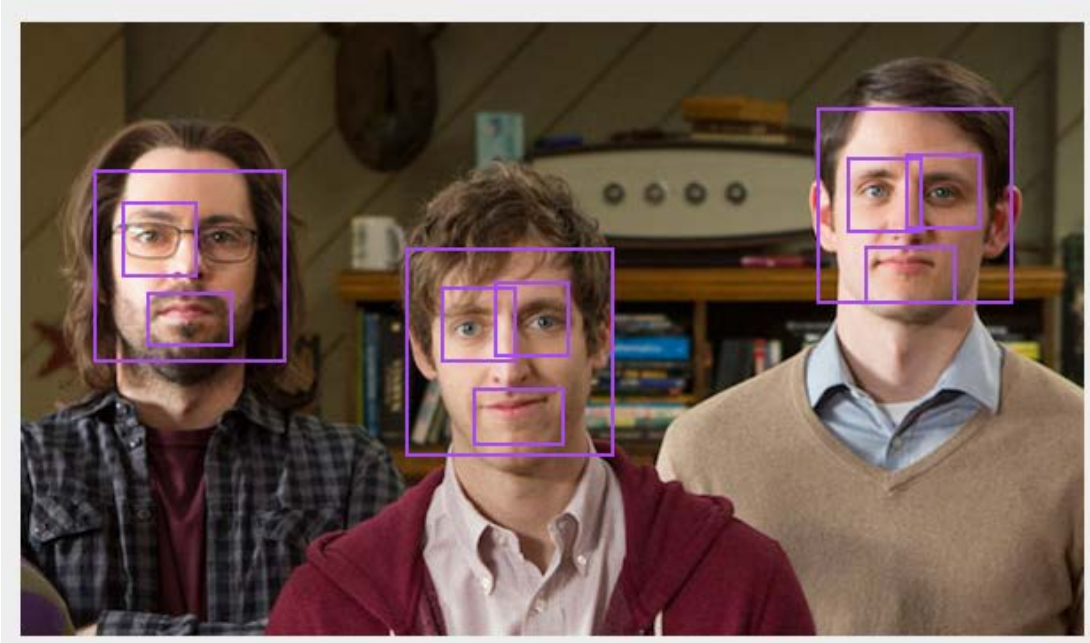
model-viewer 是谷歌自家最新的 webAR 方案，被称为具有标杆性，源码仓库是：https://github.com/google/model-viewer#augmented-reality 但在源码下载完后出现无法编译的情况。

十五、 Tracking.js

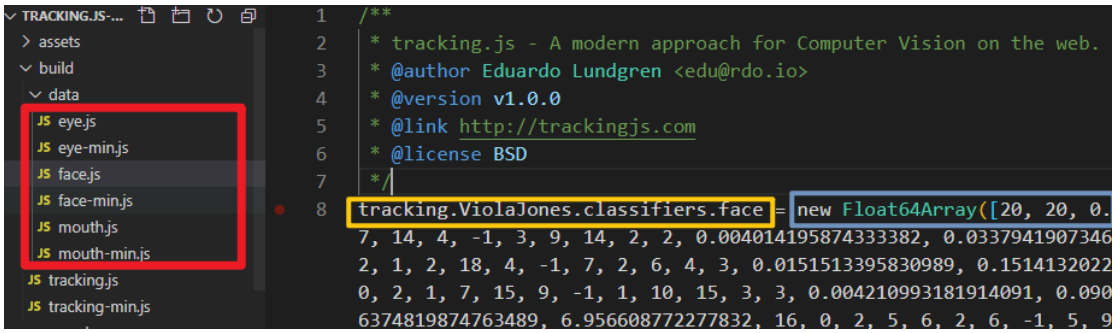
该库更多的是对于特征点的跟踪，比如面部识别、图像处理、图像特征点提取等。然而针对于任意平面的跟踪，笔者在其官网提供的库里面尚未找到蛛丝马迹，并且该库在安装依赖的时候多处报弃用甚至错误的提示。在经过一番修复之后打开本地服务器运行，发现其能够正常跑通各类演示，通过这些演示可以发觉该库确实是名副其实。它所提供开发者调用的方式也很便利。

想要实现任意平面的 AR 效果，笔者认为更改该库底层的 tracking.js 脚本的代码或仿照 eye.js、mouth.js、face.js 等脚本进行更改，写出一个关于识别任意地面特征地 plane.js 或许就能够闯过重要关卡，直抵问题地核心。尽管拥有庞大的 JavaScript 社区支持，tracking.js 却还没有 OpenCV 那么多的算法。

但是问题在于如何仿照着示例中的代码去撰写一个可以识别任意平面的“plane.js”脚本，从官网提供的代码中可以看见，其实 face.js 里面的代码是一个单精度浮点型 64 位数组，那么这样的数组具有什么特征？它又是怎么得来的？诸多问题仍等待对 SLAM 算法或是计算机视觉领域十分娴熟的人们予以解答。



[官网源码给出地面部识别示例]



[源码中面部识别所涉及的代码]

结论：

笔者所使用的开发环境是 Windows 10，测试日期为 2020 年 7 月 15 日到 7 月 16 日。通过对上述 15 个方案的测试，笔者发现能够实际测得无卡识别的 Web AR 方案有 8th-Wall 和 XR+；官方有提及相关功能点但是因为笔者测试技术原因未能见其真容的无卡识别 Web AR 方案是 zap.works、vectary、three.ar.js 和 awe.js。

通过上述结论可以得知，目前欲实现基于无卡识别的 Web AR，可以考虑使用 tracking.js，通过自己训练好的平面特征，将训练好的模型像识别人脸、识别颜色那样放入文件夹中，与 three.ar.js 或 AFrame AR 等融合（其实也可以不融合它们，直接单独使用）就能够实现基于无卡识别的的 Web AR 应用。

笔者认为，通过为 tracking.js 撰写任意地面识别的算法，外加训练好的模型，实现对任意地面的识别是可行的方案。这个方案中，难点将主要集中于识别地面算法的撰写上，该算法起码需要有和目前 tracking.js 识别人脸部同样的稳定性，即起码达到可识别任意地面的程度。至于训练好的模型，倒还容易通过大量的样本训练出来。

最后，给出笔者在测试过程中所创建的一些示例代码供读者参考和使用：https://ezharjan.github.io/WebARDemos/ ， 其中有些案例所需要用到的卡片直接在仓库里面就能找到。

参考资料：

1. https://sites.google.com/view/brown-vr-sw-review-2018/related-technology/webar-comparison-ar-js-vs-a-frame-ar

2. Three.js and AR.js Examples: https://stemkoski.github.io/AR-Examples/
3. 谷歌 WebXR 官方示例： https://www.chromestatus.com/feature/5732397976911872
4. WebXR demos: https://immersive-web.github.io/webxr-samples/
5. WebXR 技术： https://developers.google.com/web/updates/2018/06/ar-for-the-web#what
6. WebXR repository： https://github.com/immersive-web/webxr-samples
7. WebXR Hit Test Demo repository: https://github.com/immersive-web/hit-test/
8. Three.ar.js repository: https://github.com/google-ar/three.ar.js
9. Immersive Web 中的真实场景中的地面识别案例：
   https://github.com/immersive-web/real-world-geometry/blob/master/plane-detection-explainer.md
10. 制作可识别卡片： https://connected-environments.org/making/ar-playing-cards/
11. AR.js 基础：
    https://medium.com/chialab-open-source/ar-js-the-simpliest-way-to-get-cross-browser-ar-on-the-web-8f670dd45462
12. 在线卡片生成器： https://au.gmented.com/app/marker/marker.php
13. Wikitude 官网： https://www.wikitude.com
14. 方案整理： https://www.cnblogs.com/Mr147/p/13068673.html
15. Khronos Group ： https://www.khronos.org/
16. 关于 gltf 格式详解： （知乎） https://zhuanlan.zhihu.com/p/65265611 ；
    https://github.com/KhronosGroup/glTF-Tutorials/blob/master/gltfTutorial/gltfTutorial_002_BasicGltfStructure.md （官网）
17. 计算机视觉——跟踪相关的库： https://trackingjs.com/
18. Three.ar.js 在通用浏览器上无法运行后的官网解决方案指向：
    https://developers.google.com/ar/develop/web/getting-started