Google Tango vs Google ARCore vs Apple ARKit

Patrick Fehling
Hochschule für Technik und Wirtschaft Berlin, Deutschland
E-Mail: p.fehling@student.htw-berlin.de
10. Oktober 2017

***Zusammenfassung*—placeholder**

***Schlüsselbegriffe*—Augmented Reality; Google Tango; Google ARCore, Apple ARKit.**

I. Einleitung

Google Tango wurde erstmals am 3. November 2014 der Öffentlichkeit zur Verfügung gestellt [**?**]. Im Juni 2017 stellte Apple auf seiner Apple Worlwide Developers Conference iOS 11 mit ARKit vor [**?**]. Als Antwort darauf ist das Ende August 2017 erschienende Google ARCore zu verstehen, welches im Gegensatz zu Tango ohne zusätzliche Hardware auskommt [**?**].

ARCore wird weitestgehend als Nachfolger von Google Tango angesehen [**?**][**?**]. Die offizielle Aussage hierzu ist: „We've been developing the fundamental technologies that power mobile AR over the last three years with Tango, and ARCore is built on that work."[**?**] Ein weiteres Indiz für die Ablösung von Tango ist die Tatsache, dass es seit Juni keine neuen Releases der Plattform mehr gab, obwohl vorher immer mindestens monatlich eine neue Version veröffentlicht wurde [**?**].

Aus diesem Grund bin ich auch der Meinung, dass Tango ersetzt wurde und eher nach und nach Funktionalität aus der Tango SDK in ARCore umziehen werden, sofern die benötigte Hardware verfügbar gemacht wird.

In dieser Arbeit soll nun geklärt werden, welche Unterschiede zwischen Tango und dem ARCore bestehen, also welche Verluste man dadurch einbüßt, und wie konkurrenzfähig dies zum Apple ARKit ist.

II. Google Tango

Google Tango ist eine Plattform für Augmented Reality und Computer Vision für Android. Per Motion Tracking, Gyroskop und Beschleunigssensor ermittelt das Gerät seine Position im Raum. Über infrarotes Structured Light und Time-of-Flight-Messungen, sowie Stereo-Kameras werden Tiefenmessungen durchgeführt. Dadurch kann der Raum gescannt und in einer Punktwolke, die sog. „Tango Point Cloud", wiedergegeben werden. Diese kann dann z.B. dazu verwendet werden, virtuelle Objekte im realen Raum zu platzieren oder die reale Welt virtuell abzubilden. Für all dies wird spezielle zusätzliche Hardware (z.B. IR-Projektor, Infrarotsensor) im Gerät benötigt.[**?**]

Nachdem ich mich in meiner letzten Arbeit theoretisch mit den Konzepten von Google Tango auseinandergesetzt, hatte ich nun die Möglichkeit auch praktisch mit Google Tango zu arbeiten. Dazu nutzte ich das Lenovo Phab 2 Pro, das erste Tango-Gerät für Endverbraucher. Für den Einstieg bietet Google eine „HowTos" für Unity an. Folgendes wurde dabei umgesetzt:

1) **Platzieren einer Kugel:** Nach dem Start der App wird die nächste beste Position zum Platzieren der Kugel gesucht und dort wird sie platziert. Anschließend kann man mit dem Gerät um diese herumlaufen.
2) **Platzieren von Objekten bei Nutzereingabe:** Per Tap auf dem Bildschirm wird der Schnittpunkt zu auf dem berührten Pixel liegenden Oberfläche ermittelt und auf diesem wird ein Objekt platziert (hier: eine animierte Katze).
3) **Scan eines Raums:** Es wird mit dem Gerät der Raum gescannt. Währenddessen wird ein Mesh des Raumes erstellt, welches als obj.Datei exportiert werden kann.
4) **Visualisierung der Punktwolke:** Anstatt das normale Kamerabild oder eine fremde virtuelle Welt zu sehen, wird die reale Welt als Punktwolke, sowie sie vom Gerät "gesehen"wird dargestellt.
5) **AreaLearning:** Bei der Applikation lassen sich Marken im Raum verteilen und speichern. Nach einem Neustart, werden diese Marken in etwa am gleichen Ort wieder platziert.

Ein besonderes Feature von Tango ist das „Area Learning". Dabei wird ein „Gedächtnis" der Umgebung anhand von Landmarken aufgebaut. Diese werden in einer Area Description File (ADF) Verliert das Gerät die Orientierung findet es über das Area Learning wieder zurück.[**?**]

Auf die gespeicherten Landmarken hat man jedoch keinen direkten Zugriff. Die Tango Point Cloud kann jedoch dadurch angepasst bzw. aktualisiert werden. Wenn sich das Gerät mithilfe der Landmarken lokalisiert hat, wird das Koordinatensystem der Punktwolke aktualisiert. Somit können Koordinaten von z.B. platzierten virtuellen Objekten persistiert werden und erscheinen zu einem späteren Zeitpunkt (z.B. nach dem Neustart der Anwendung) an derselben Stelle.

Dies wurde anhand einer Beispiel-App von Google praktisch getestet. Die Beispiele befinden sich direkt im Unity-Package der TangoSDK, welches in Unity importiert werden muss, um die Tango-Funktionen zu nutzen. Die Applikation heißt „Area Learning" und ermöglicht das Platzieren von verschiedenfarbigen Markern im Raum sowie das Persistieren dieser Informationen. Dabei wird eine ADF erstellt und zusätzlich werden die Koordinaten, die Ausrichtung der Marker und die

Farbe in einer XML-Datei gespeichert. Nach dem Neustart der Anwendung und der erfolgreichen Lokalisierung wird die Datei eingelesen und die Marker werden erneut platziert.
Wenn zunächst die ADF erstellt und gespeichert wird und anschließend die Marker platziert und gespeichert werden, funktioniert der eben beschriebene Ablauf problemlos. Falls dies jedoch gleichzeitig passiert und die „Welt“ wiederaufgebaut wird, sind alle Marker um ca. 90°gedreht. Diese Verhalten wird auch in den Issues auf Github angesprochen[**?**].
Die Applikation lies sich auch dahingehend erweitern, dass zusätzlich Textinformationen gespeichert werden, welche am Marker angezeigt werden. Dies könnte z.B. eine Grundlage für einen textbasierten Museumsguide sein. Des Weiteren wurde die Möglichkeit des Entfernens von ADFs implementiert.
Neben Unity unterstützt Tango auch Java und C. Da Unity eine Game Engine ist und daher auf einer relative hohen Abstraktionsebene arbeitet, schaute ich auch in die Java API hinein. Zu dieser werden von Google keine Tutorials geliefert, jedoch haben sie auf Github eine Reihe von Beispiel-Applikationen, sowohl für Java (J) [**?**] als auch für Unity (U) [**?**]:

- **Hello Area Description / Area Description Management (J, U):** Erstellen von Speichern von ADFs. Die App zeigt zusätzlich wann man im ADF lokalisiert ist und wann nicht.
- **(Simple) Augmented Reality (J, U):** Platziert Mond und Erde an die nächst beste Position. Diese gibt es einmal als reine OpenGL ES Applikation und einmal mit der Rajawali Engine.
- **Find Floor (U):** Sucht in der aktuell gesehen Szene die niedrigste Ebene des Raumes (Boden).
- **Floor Planner (J):** Erstellt den Grundriss des gescannten Gebiets.
- **Green Screen (J):** Simuliert einen Greenscreen mithilfe der Tiefendaten (hintere Bereiche werden ausgeblendet).
- **Mesh Builder (J, U):** Gleiche Funktionsweise wie der Raumscan in Unity.
- **Model Correspondance (J):** Platziert ein Haus zwischen vier vom Nutzer gesetzten Punkten. Die größe des Hause hängt von der mit den Punkten markierten Fläche ab.
- **Motion Tracking (J, U)** Zeigt eine virtuelle Welt, in der man sich über Motion Tracking bewegen kann.
- **Occlusion (J):** Per Tap auf dem Bildschirm wird eine Erde platziert, welche von anderen realen Objekten verdeckt werden kann.
- **Point Cloud (J, U):** Visualisiert die Punktwolke. Der Nutzer kann zwischen verschiedenen Perspektiven wählen. Dabei handelt es sich um eine komplexere Variante als die von den Unity-„HowTos“.
- **Point To Point (J, U):** Der Nutzer setzt zwei Punkte per Tap auf dem Bildschirm. Die Strecke zwischen den beiden Punkten wird visualisiert und die Länge wird berechnet.

Mit dem Floor Planner, der Model Correspondance App und dem Occlusion-Beispiel haben die Java-Beispiele werden interessante Funktionen präsentiert. Was das Area Learning anging, war Unity jedoch die Grundlage mit dem geringeren Aufwand. Aufgrund der verschiedenen Programmiersprachen bzw. „Entwicklungswelten“ unterscheidet sich die Entwicklung deutlich. Beim Erhalten der Daten und deren Verarbeitung sind jedoch Gemeinsamkeiten zu erkennen.

III. Google ARCore

Google ARCore ist ebenfalls eine Augmented-Reality-Plattform. Offiziell ist es nur auf den Google Pixel Phones und dem Samsung Galaxy S8 unterstützt. Nach dem Verlassen des Preview-Status sollen noch weitere Geräte folgen. Es basiert auf drei fundamentalen Konzepten [**?**]:

1) **Motion tracking:** Wie bei Tango wird per Feature Points im gesehenen Bild die Position und Ausrichtung des Geräts im Raum ermittelt. Daten aus dem Gyroskop und Beschleunigungssensor (IMU) des Telefons werden hierbei ebenfalls mit den Bilddaten kombiniert. [**?**]
   Problem sind bei ruckartigen Bewegungen zu erwarten. Google Tango löste dieses Probleme mit dem Area Learning.
2) **Environmental understanding:** Durch Analyse der Feature Points werden flache Oberflächen erkannt und können z.B. mit Objekten bestückt werden. [**?**]
   In Google Tango bietet z.B. die TangoPointCloud in der Unity-API per `findPlane`-Methode eine sehr ähnliche Funktionalität.
3) **Light estimation:** Die reale Beleuchtung wird analysiert und ARCore stellt diese Informationen zur Verfügung, sodass virtuelle Objekte durch korrekte Beleuchtung realistischer aussehen. [**?**]
   Ein solches Feature gibt es in der Google Tango Plattform nicht.

ARCore baut laut Google auf der Arbeit von Tango auf. Der Vergleich in Tabelle I zeigt, dass sich beide sehr weit voneinander entfernen, was zum einen daran liegt, das beide mit unterschiedlichen Daten arbeiten. Zum Anderen könnte dies aber auch am Apple ARKit liegen, was später noch deutlicher wird.

ARCore	**Tango**
Anchor	
Config	TangoConfig
Frame	TangoImageBuffer
HitResult	
LightEstimate	
Plane	
PlaneHitResult	
PointCloud	TangoPointCloud
PointCloudHitResult	
Pose	TangoPoseData
Session	Tango

Tabelle I: Gegenüberstellung der Schnittstellen von Google ARCore und Google Tango

IV. Apple ARKit

Das ARKit von Apple ist ein Augmented-Reality-Framework, welches mit iOS 11 veröffentlicht wurde [**?**].

ARKit und ARCore sind ihrem Aufbau sehr ähnlich, sowohl in den Konzepten, als auch in den Schnittstellen. In Tabelle II ist dies verdeutlicht.

ARKit	**ARCore**
Konzepte	
Visual Inertial Odometry	Motion Tracking
Scene Understanding	Environmental Understanding
Light Estimation	Light Estimation
Schnittstellen	
ARAnchor	Anchor
ARConfiguration	Config
ARFrame	Frame
ARHitTestResult	HitResult
ARLightEstimate / ARDirectionalLightEstimate	LightEstimate
ARPlaneAnchor	Plane
ARSession	Session
ARFaceAnchor	—
ARCamera	—

Tabelle II: Gegenüberstellung von Apple ARKit und Google ARCore

Womit das ARKit deutlich hervorsticht ist die „Face-Based AR Experience“. Mithilfe der „TrueDepth Camera“ als Front-Kamera des iPhones kann die Position und das Aussehen des Gesichts erfasst werden und z.B. auf ein virtuelles Gesicht übertragen werden. Ein weiterer Punkt ist das komplette ausblenden des Hintergrunds.[**?**]
Die Technologie dahinter ist der Tango-Hardware sehr ähnlich. Es gibt einen Infrarotsensor und einen „Dot-Projector“, wodurch die Tiefe der Szene, also des Gesichts, ermittelt wird. [**?**]

V. Fazit

ARCore ist von außen deutlich näher am ARKit als an Google Tango.