

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

15. Oktober 2017

Zusammenfassung—Google Tango war die erste markerlose „Augmented Reality“-Plattform für Smartphones. Das Apple ARKit und das ARCore aus dem eigenen Hause stellen große Konkurrenz dar, da sie im Gegensatz zu Tango keine extra Hardware benötigt. Am Beispiel von ARKit wird gezeigt was eine aktuelles Smartphone in AR leisten kann. Ein besonderer Fokus liegt dabei auf dem Area Learning, welches noch ein Alleinstellungsmerkmal von Tango zu sein scheint.

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

I. EINLEITUNG

Google Tango wurde erstmals am 3. November 2014 der Öffentlichkeit zur Verfügung gestellt [1]. Im Juni 2017 stellte Apple auf seiner Apple Worldwide Developers Conference iOS 11 mit ARKit vor [2]. Als Antwort darauf ist das Ende August 2017 erschienende Google ARCore zu verstehen, welches im Gegensatz zu Tango ohne zusätzliche Hardware auskommt [3].

ARCore wird weitestgehend als Nachfolger von Google Tango angesehen [4][5]. Eine offizielle Aussage von Google 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.“[3] Dabei wird nicht direkt vom Ende der Tango-Plattform gesprochen. Ein weiteres starkes Indiz für die Ablösung von Tango ist jedoch die Tatsache, dass es seit Juni keine neuen Releases der Plattform mehr gab, obwohl vorher immer mindestens monatlich eine neue Version veröffentlicht wurde [1].

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.

Die nötige Hardware für ARCore steht mir leider nicht zur Verfügung. Außerdem befindet sich ARCore noch im Preview-Status. Aufgrund der starken Ähnlichkeit von ARCore und ARKit sollte ein Vergleich zwischen Tango und ARKit hier ausreichen.

II. GOOGLE TANGO

Google Tango ist eine Plattform für Augmented Reality und Computer Vision für das Android-Betriebssystem. Per Motion Tracking, Gyroskop und Beschleunigungssensor 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. [6]

Nachdem ich mich in meiner letzten Arbeit theoretisch mit den Konzepten von Google Tango auseinandergesetzt hatte, 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 Reihe von „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) gespeichert. Verliert das Gerät die Orientierung findet es über das Area Learning, also über einen Abgleich mit den Landmarken, wieder zurück.[6]

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 Koordinaten-

system 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 und leicht verschoben. Dieses Verhalten wird auch in einem Issue auf dem Github-Repository angesprochen[7].

Ausgehend von dieser Applikation wurde eine Anwendung zu Erstellung minimalistischer und textbasierter Museumsguides entwickelt. Nach dem Start der Applikation kommt man in das Hauptmenü. Dort wählt man eine ADF aus und drückt „Start“ oder man startet per Button „New Area Description“ die Anwendung mit einer neuen ADF. Weiterhin kann man die ausgewählte ADF per „Delete“-Button löschen. Eine Checkbox in der linken unteren Ecke schaltet den Learning Mode ein. In diesem Modus wird die ausgewählte ADF beim Herumlaufen im Raum ggf. erweitert.

Nach dem Start erscheint erst einmal auf dem Bildschirm der folgende Text „Walk around to relocalize“. Im Fall einer neu erstellten ADF sollte dieser Text sehr schnell verschwinden. Bei einer vorher ausgewählten ADF ist dies stark abhängig von der Umgebung. Nach der Lokalisierung erscheint das Kamerabild und ein Menü in der rechten unteren Ecke (siehe Abbildung 1). Dort kann zwischen drei verschiedenen Farben für die Marker ausgewählt werden. Per Touch auf eine beliebige Stelle außerhalb der UI wird die Fläche am berührten Pixel ermittelt und an dieser Stelle wird ein Marker platziert. Des Weiteren öffnet sich die Systemtastatur. Hier hat der Nutzer die Möglichkeit dem Marker einen Namen zu vergeben.

Der platzierte Marker kann ebenfalls berührt werden. Anschließend erscheinen zusätzliche UI-Elemente, welche in Abbildung 1 zu sehen sind. Unter dem Marker ist ein Button zum Löschen des Markers. Über dem Marker sieht man den Namen des Markers und rechts erscheint zunächst ein rechteckiger leerer Kasten. Wenn dieser berührt wird öffnet sich erneut die Systemtastatur und der Nutzer kann einen Info-Text o.ä. zum Marker eingeben.

Neben Unity unterstützt Tango auch Java und C. Da Unity eine Game Engine ist und daher auf einer relativ hohen Abstraktionsebene arbeitet, schaute ich auch in die Java API

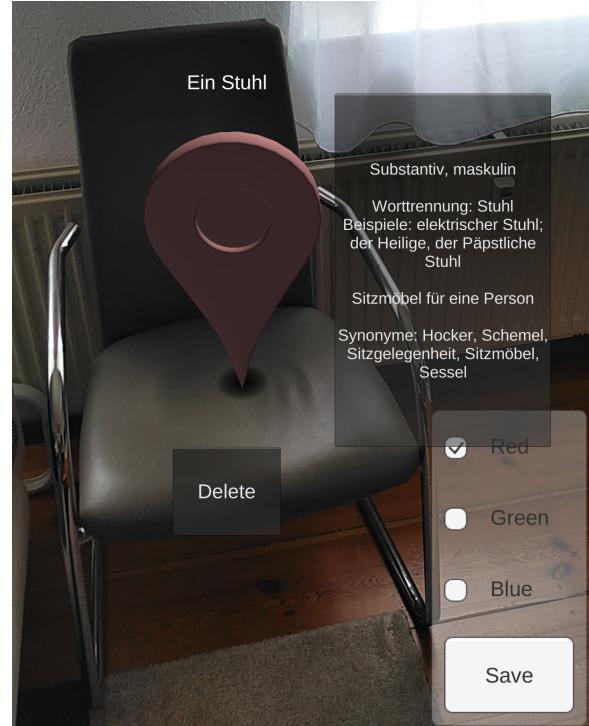


Abbildung 1: Ein Marker in der App „Guide Creator“ mit zusätzlichen Informationen

hinein. Zu dieser werden von Google keine Tutorials geliefert, jedoch haben sie auf Github eine Reihe von Beispiel-Applikationen, sowohl für Java (J) [8] als auch für Unity (U) [9]:

- **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 gesehenen Szene die niedrigste Ebene des Raumes (i.d.R. der Boden).
- **Floor Planner (J):** Erstellt den Grundriss des gescannten Gebiets.
- **Green Screen (J):** Simuliert einen Greenscreen mithilfe der Tiefendaten, d.h. hintere (tieferen) 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 bieten die Java-Beispiele Funktionalitäten, die man in den Unity-Beispielen nicht findet. Dies heißt natürlich nicht automatisch, dass diese nicht in Unity umsetzbar wären. 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 eine Vielzahl von Geräten folgen. Im Gegensatz zu Tango wird keine zusätzliche Hardwärde benötigt, d.h. lediglich Kamera, das Gyroskop und der Beschleunigungssensor werden verwendet, also Komponenten, die in jedem aktuellem Smartphone zu finden sind. Dabei basiert es auf drei fundamentalen Konzepten [10]:

- 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. [11] 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. [11] In Google Tango bietet z.B. die TangoPointCloud in der Unity-API per `findPlane`-Methode eine sehr ähnliche Funktionalität. Auch ARCore stellt eine Punktwolke zur Verfügung.
- 3) **Light estimation:** Die reale Beleuchtung wird analysiert und ARCore stellt diese Informationen zur Verfügung, sodass virtuelle Objekte durch korrekte Beleuchtung realistischer aussehen. [11] 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.

Das Tango SDK ist komplexer und teilweise auch technischer aufgebaut, wobei bei ARCore die Einfachheit und der Anwendungszweck mehr im Vordergrund liegen. Wie viele Features genau dadurch verloren gehen ist ohne ARCore testen zu können schwer einzuschätzen. Da in den Dokumentation nichts zu Area Learning oder ähnlichem steht, fehlt dies vermutlich hier.

Schon in Vuforia werden jedoch in Markern visuelle Features verwendet um den Marker zu erkennen [6]. Die gespeicherten Landmarken werden auch zum Großteil aus diesen bestehen, da die Lokalisierung bei schlechten Lichtverhältnissen teilweise nicht funktioniert. Also kann, wenn es noch nicht vorhanden ist, dies durchaus noch bis zum finalen Release implementiert werden.

ARCore	Tango (Ausschnitt)
Anchor	—
Config	TangoConfig
Frame	TangoImageBuffer
HitResult	—
LightEstimate	—
Plane	—
PlaneHitResult	TS.IntersectionPointPlaneModelPair
PointCloud	TangoPointCloudData/ TangoPointCloudManager
PointCloudHitResult	—
Pose	TangoPoseData
Session	Tango
—	TangoAreaDescriptinMetaDta
—	TangoCameraIntrinsics
Session	TangoCameraNativeLoader
—	TangoCoordinateFramePair
—	TangoEvent
—	TangoTextureCameraPreview
—	TangoXyzljData

Tabelle I: Gegenüberstellung der Schnittstellen von Google ARCore[12] und Google Tango[13]
Legende: TS = TangoSupport

IV. APPLE ARKIT

Das ARKit von Apple ist ein Augmented-Reality-Framework, welches mit iOS 11 veröffentlicht wurde [14]. ARKit und ARCore sind in 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[15] und Google ARCore[12]

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.[16]

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. [17]

Ein beliebtes Genre für AR-Apps ist das Messen der Länge oder Größe von realen Objekten. Dazu gibt es eine Vielzahl von Apps im Apple Appstore, z.B. die aktuell bewertesten „AirMeasure“ oder „3-in-1 Ruler“. Google ist diesem Trend zuvorgekommen und liefert mit dem Tango-Gerät eine eigene vorinstallierte App namens „Measure“ mit. Mit diesen Applikationen lässt sich aber auch gut die Präzision der beiden Plattformen testen. Die Ergebnisse der Messungen sind in Tabelle III zu sehen.

Google Measure unterscheidet sich um wenige Zentimeter von den realen Maßen. Die Differenzen entstehen zum Großteil aufgrund der Benutzeroberfläche der App. In der Mitte der App befindet sich ein Punkt mit dem die Endpunkte der Strecke markiert werden. Aufgrund der Größe dieses Punktes wird allerdings das präzise Platzieren der Endpunkte erschwert.

Die ARKit Apps sind gerade bei kleineren Strecken ziemlich präzise. Bei der langen Strecken wie z.B. der Länge des Türrahmens ist die Abweichung jedoch ziemlich groß. Eine der Apps war nicht in der Lage auf vertikalen Flächen zu messen, weshalb die Messung des Türrahmens dort gar nicht möglich war. Allerdings könnte das auch den Defizit der anderen App bei dieser Strecke erklären, was bedeuten würde, dass das ARKit nur horizontale Flächen erkennt. Des Weiteren ist die Bedienung der Apps etwas schwieriger, da das Tracking teilweise verloren geht. Dabei kam eine Nachricht, dass man für bessere Lichtverhältnisse sorgen solle. Ich denke aber an dieser Stelle war wieder das Problem, dass es keine horizontale Ebene gab.

Das zu messende Objekt muss vor der Messung aus mehreren Positionen abgefilmt werden. Letzteres ist ein starkes Indiz dafür, dass beim ARKit Stereoskopie zum Einsatz kommt, d.h. die Tiefe in den Bildern wird aus mehreren Kamerabildern berechnet. Es wird eine zweite Kamera simuliert. Sobald eine horizontale Ebene erfasst wurde und man immer solch eine Ebene im Bild war, war das Tracking in den Tests störungsfrei. Ein platziertes Objekt kann in Abbildung 2 gesehen werden. Das Objekt steht dabei direkt auf dem Boden und bleibt bei Bewegungen an der gleichen Stelle. Im Vergleich zu Google Tango ist dies mindestens genauso gut, wenn nicht sogar etwas stabiler.

Beim Probieren mehrerer Apps war keine dabei, die es ermöglicht platzierte Objekte an der Stelle zu speichern. Aus diesem Grund vermute ich, dass ein Feature wie das Area Learning im ARKit nicht gibt. Auch in den Schnittstellendokumentationen habe ich dazu nichts gefunden.

	Tisch	Türrahmen	Smartphone
Zollstock	100x60	199x83	14,5x7
Goole Measure	98x58	200x84	13x6
AirMeasure	99,3x58,4	190x80,9	14,2x7,8
3-in-1 Ruler	98,8x59,3	XXXx80,5	13,7x6,7

Tabelle III: Tests von Apps, die reale Objekte abmessen (Maße in cm)



Abbildung 2: Virtuelles Objekt wird in die reale Welt projiziert (ARKit)

V. FAZIT

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

Ich denke es ist nur eine Frage der Zeit bis diese Technologie auch in der hinteren Kamera eingebaut wird. Die zusätzliche Hardware stellt für Apple-Produkte durch die geringe Produktvielfalt ein geringeres Problem dar als bei Android-Produkten, wo verschiedene Hersteller erst nachziehen müssen. Die Vergangenheit zeigt jedoch, dass eine populäre Technologie sich schnell durchsetzt, damit die Hersteller konkurrenzfähig bleiben. Die Frage ist jedoch, welchen Nutzen Endverbraucher aus Augmented Reality auf dem Smartphone ziehen können. ARCore und ARKit werden durch ihre geringen Kosten genau diese Frage beantworten.

LITERATUR

- [1] Tango SDK Release Notes. <https://developers.google.com/tango/release-notes>. Zugriff: 18.09.2017.
- [2] iOS 11 brings powerful new features to iPhone and iPad this fall. <https://www.apple.com/newsroom/2017/06/ios-11-brings-new-features-to-iphone-and-ipad-this-fall/>. Zugriff: 18.09.2017.

- [3] Dave Burke. ARToolKit. <https://www.blog.google/products/google-vr/arcore-augmented-reality-android-scale/>. Zugriff: 18.09.2017.
- [4] Sean Hollister. ARCore is Google's Tango replacement. Can it catch Apple? <https://www.cnet.com/news/google-tango-dead-arcore-arkit-apple/>. Zugriff: 18.09.2017.
- [5] Jan-Keno Janssen. Googles Augmented Reality: Tango ist tot, es lebe ARCore. <https://www.heise.de/newstickermeldung/Googles-Augmented-Reality-Tango-ist-tot-es-lebe-ARCore-3817226.html>. Zugriff: 18.09.2017.
- [6] Patrick Fehling. PTC Vuforia vs. Google Tango. In *Aktuelle Entwicklungen der Angewandten Informatik - Begleitband zum Seminar im Masterstudiengang Angewandte Informatik an der HTW Berlin, Wintersemester 2016/17*, page 20, 2017. unveröffentlicht.
- [7] Bug with AreaLearning example - Markers offset when saving. <https://github.com/googlesamples/tango-examples-unity/issues/100>. Zugriff: 08.10.2017.
- [8] Example projects for Project Tango Java API. <https://github.com/googlesamples/tango-examples-java/>. Zugriff: 08.10.2017.
- [9] Project Tango UnitySDK Example Projects. <https://github.com/googlesamples/tango-examples-unity>. Zugriff: 08.10.2017.
- [10] ARCore Overview. <https://developers.google.com/ar/discover/>. Zugriff: 18.09.2016.
- [11] Fundamental Concepts. <https://developers.google.com/ar/discover/concepts>. Zugriff: 18.09.2017.
- [12] com.google.ar.core. <https://developers.google.com/ar/reference/java/com/google/ar/core/package-summary>. Zugriff: 15.10.2017.
- [13] Tango Java API. <https://developers.google.com/tango/apis/java/reference/>. Zugriff: 15.10.2017.
- [14] Introducing ARKit. <https://developer.apple.com/arkit/>. Zugriff: 19.09.2017.
- [15] ARKit. <https://developer.apple.com/documentation/arkit>. Zugriff: 15.10.2017.
- [16] iPhone X. <https://www.apple.com/iphone-x/>. Zugriff: 19.09.2017.
- [17] iPhone X - Design and Display. <https://www.apple.com/iphone-x/#design>. Zugriff: 19.09.2017.