Augmented Reality APIs für Android

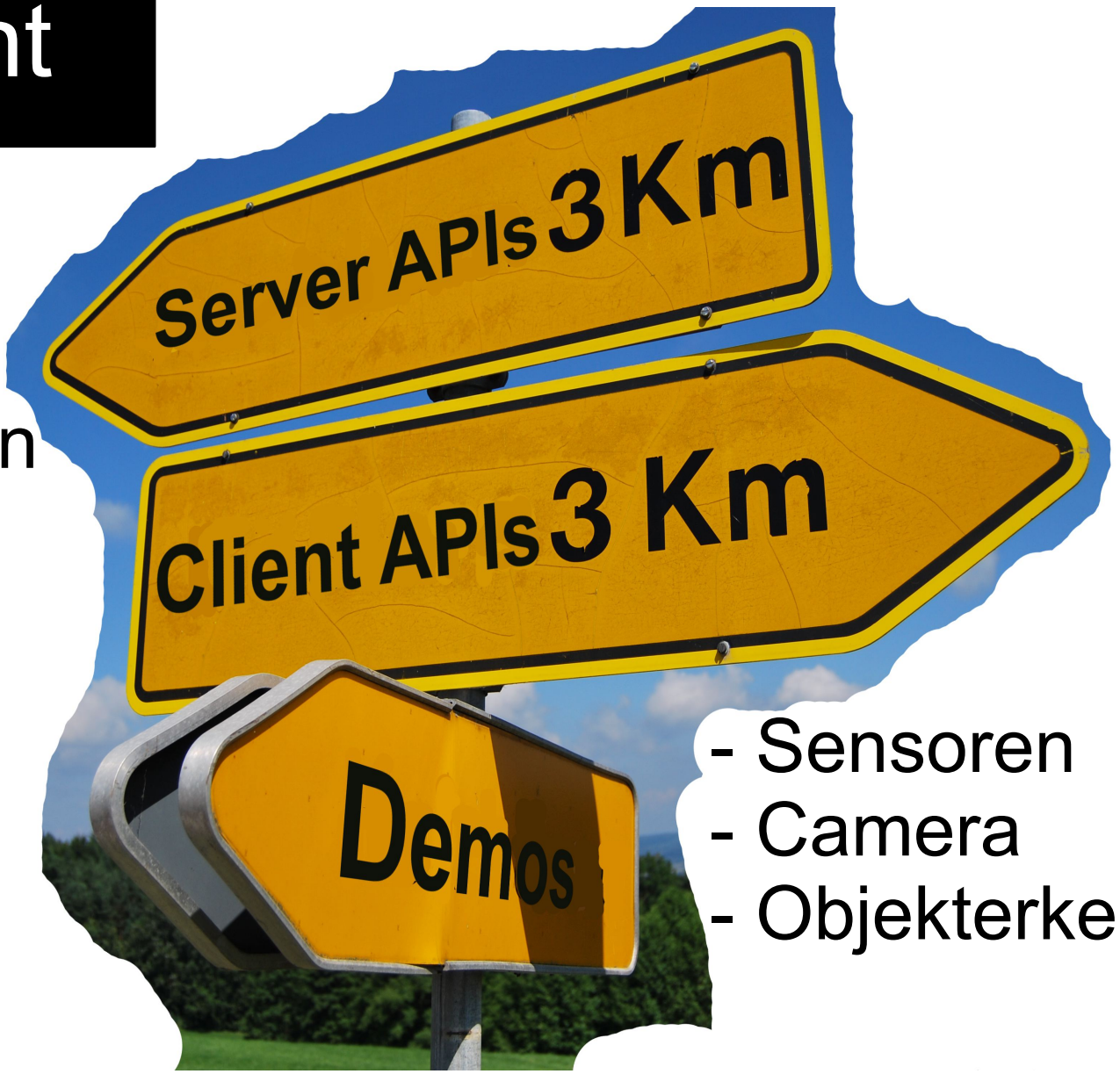
Dr.-Ing. Johannes Leebmann

- Möglichkeiten für Entwickler
- Momentane Grenzen

Java User Group Darmstadt 18.08.2010;

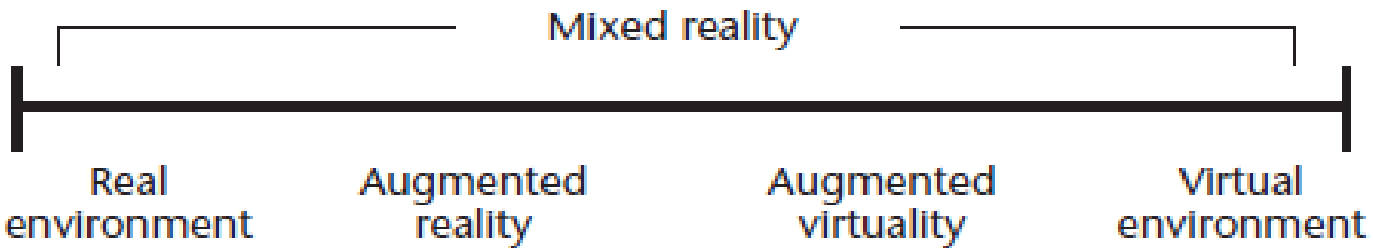
Übersicht

- Varianten
- Architekturen
- Formate



- Sensoren
- Camera
- Objekterkennung

AR Varianten



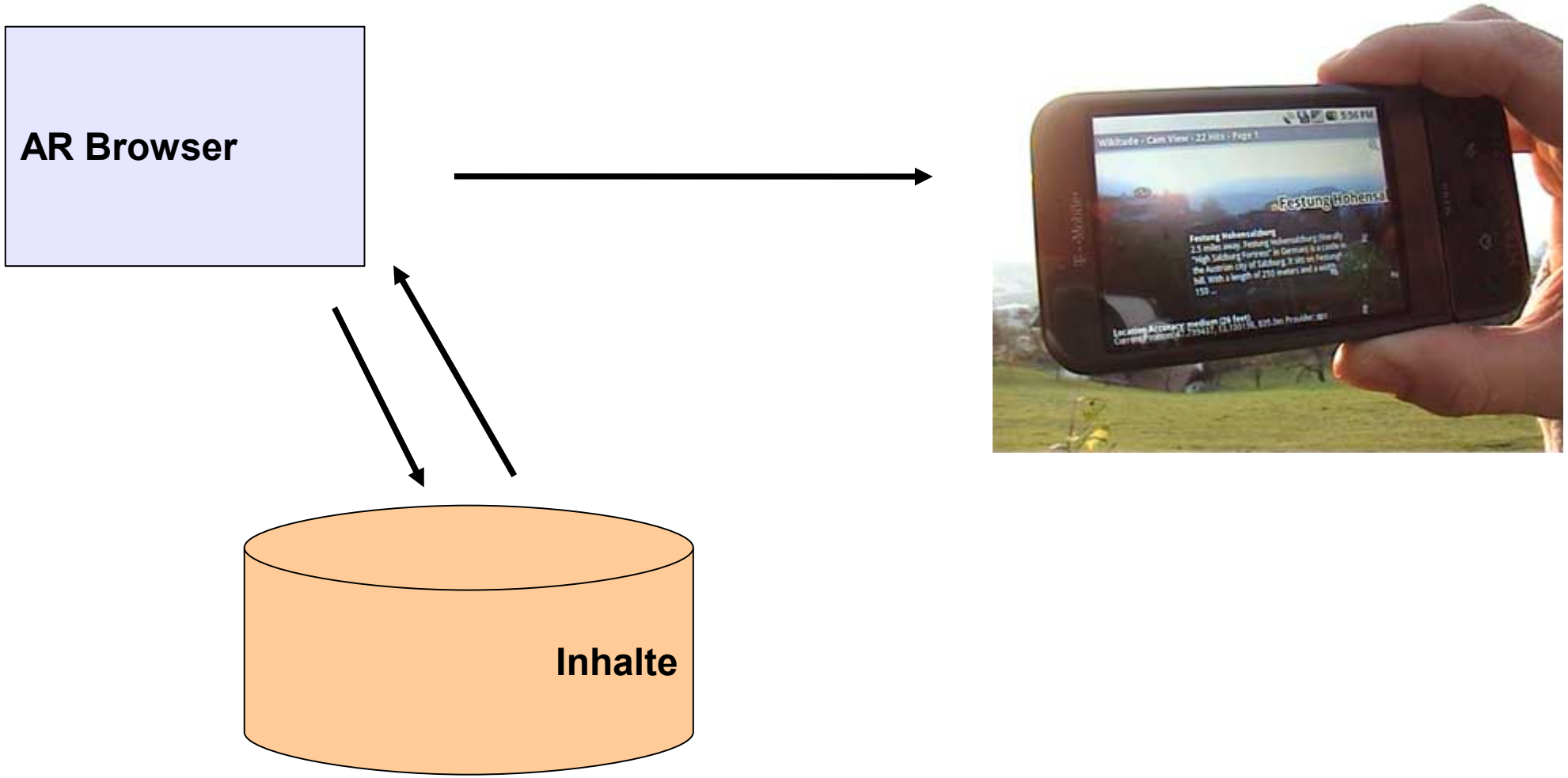
Detect Defined Markers

Detect Training Data

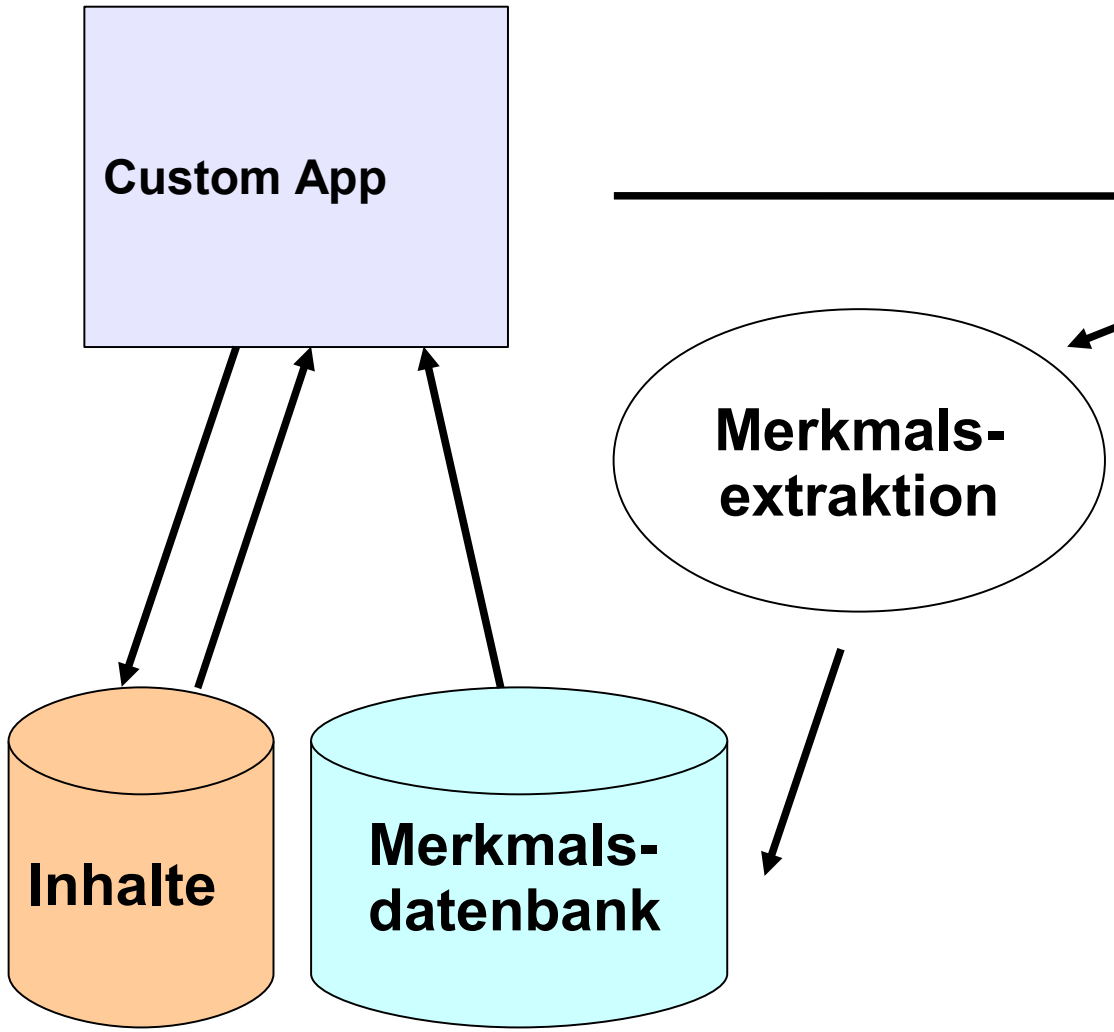
Azuma et. al. 2001: „Anteil an virtueller Welt“

„Image Understanding“

Architekturen: 1 AR Browser



Architekturen: 2 Custom AR



Interaktion und Situationsverstehen:
- **Traffic Sign Recognition**
- **Spiele**
- **Bildsuche im Internet**

Serverseitiges API

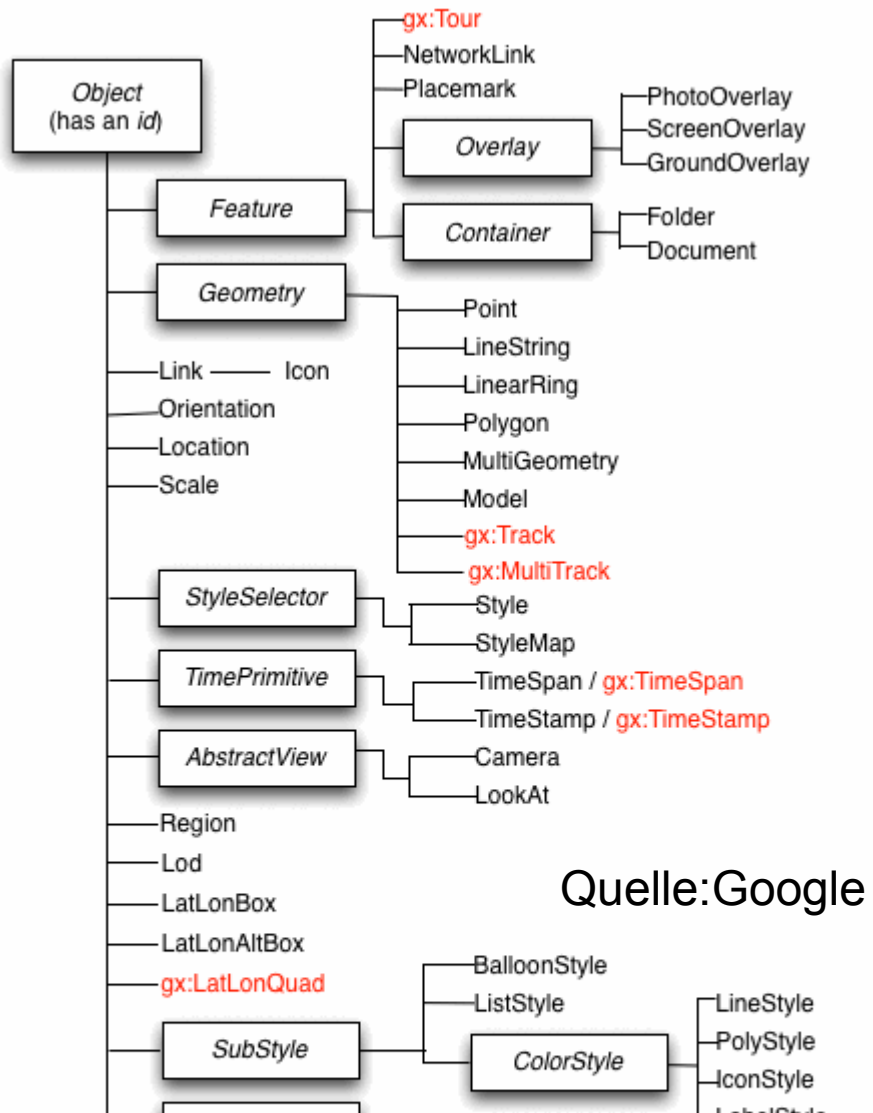
- Definition von **WebService Endpoints**
- Datenbanken **für POIs**
- www.mixare.org ermöglicht CustomClient
- **Formate:** JSON, KML, ARML, XML, Google Buzz, Twitter

Serverseitige APIs

Angabe von Points of Interest **POI**
z.B. Open Geospatial Consortium
Standard **KML**

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
  <Placemark>
    <name>Zürich</name>
    <description>Zürich</description>
    <Point>
      <coordinates>8.55,47.3666667,0</coordinates>
    </Point>
  </Placemark>
</Document>
</kml>
```

ARML erweitert um Telefonnummern,
Icons, Attachments, URLs



Quelle:Google

Clientseitige APIs **OnBoard**

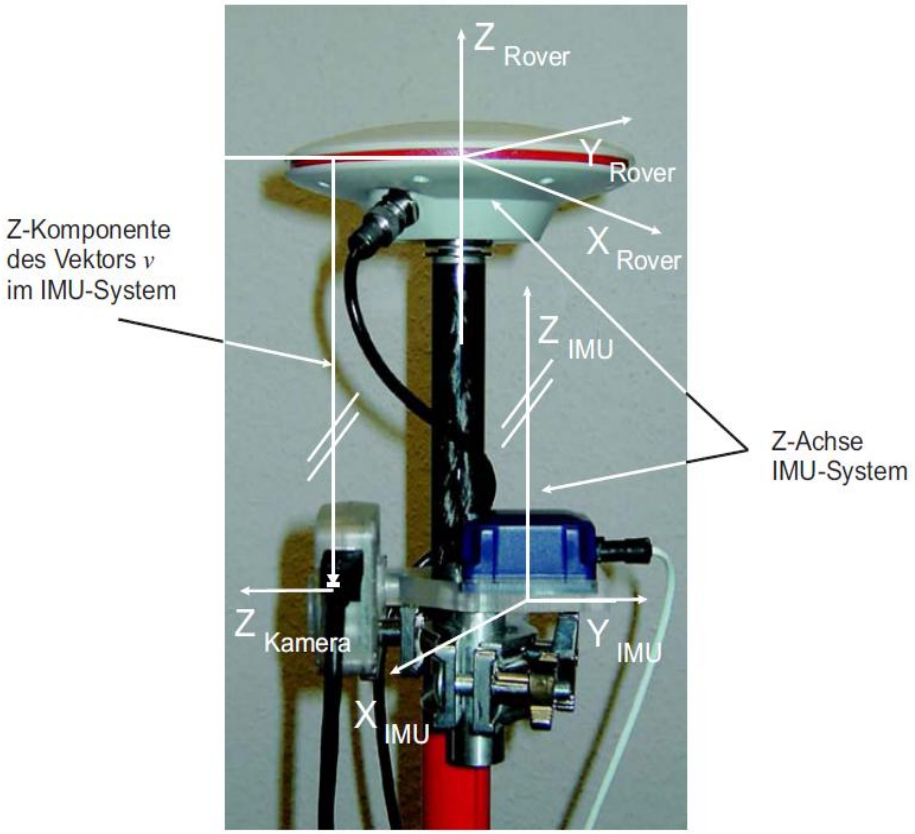
Package ***hardware:*** alle Geräte, die nicht auf jedem Android verfügbar sind.

- Camera
- GeomagneticField
- „Sensor“

Package ***location:***

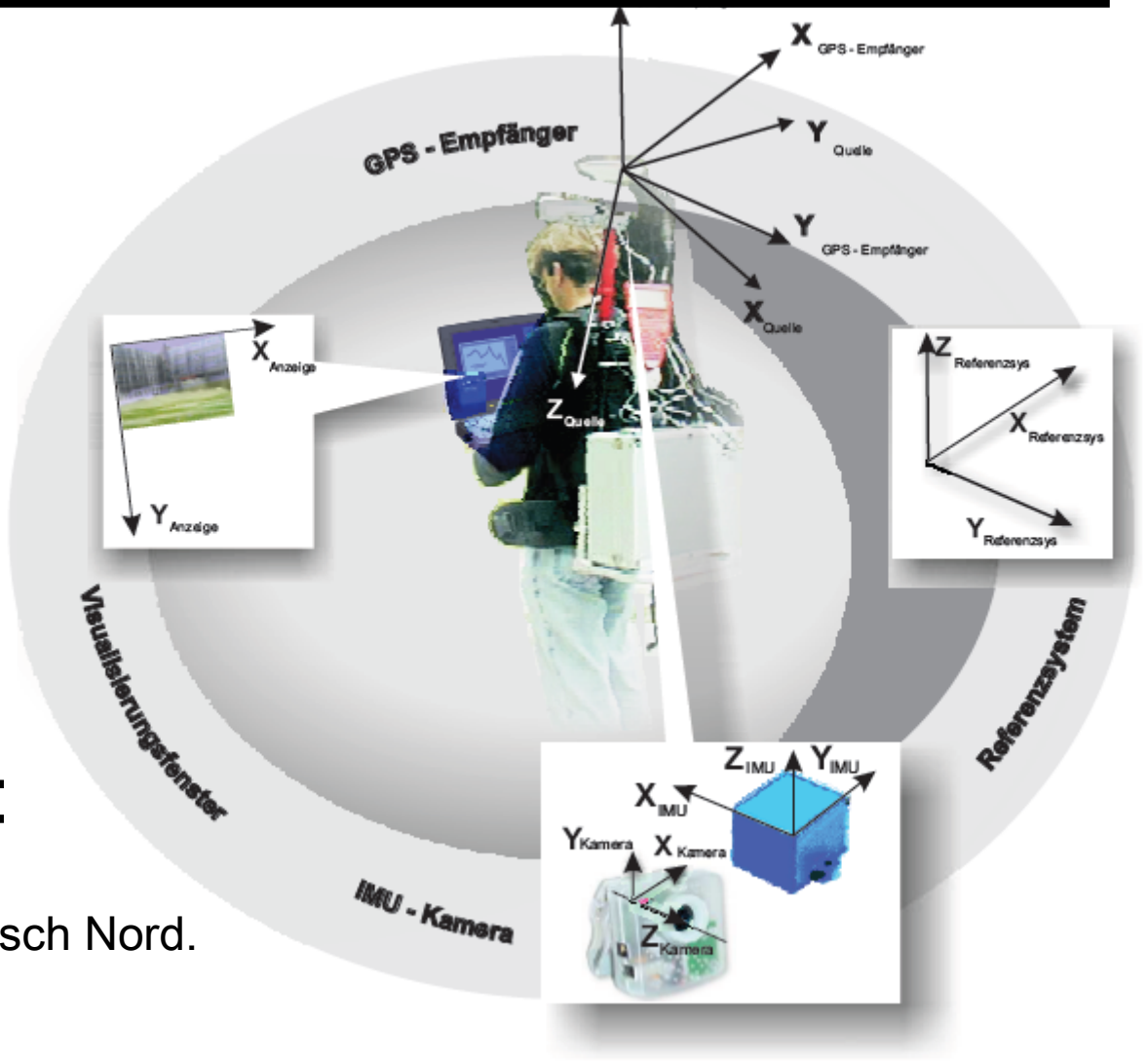
- LocationManager
- Location

Zusammenführen der Messungen



$$\mathrm{T}^{Anzeige}_{Referenzsys.}\mathrm{x}_{Referenzsys.} = \mathrm{P}^{Anzeige}_{Kamera}\mathrm{T}^{Kamera}_{IMU}\mathrm{T}^{IMU}_{Quelle}\mathrm{T}^{Quelle}_{GPS-Empfänger}\mathrm{T}^{GPS-Empfänger}_{Referenzsys.}\mathrm{x}_{Referenzsys.}$$

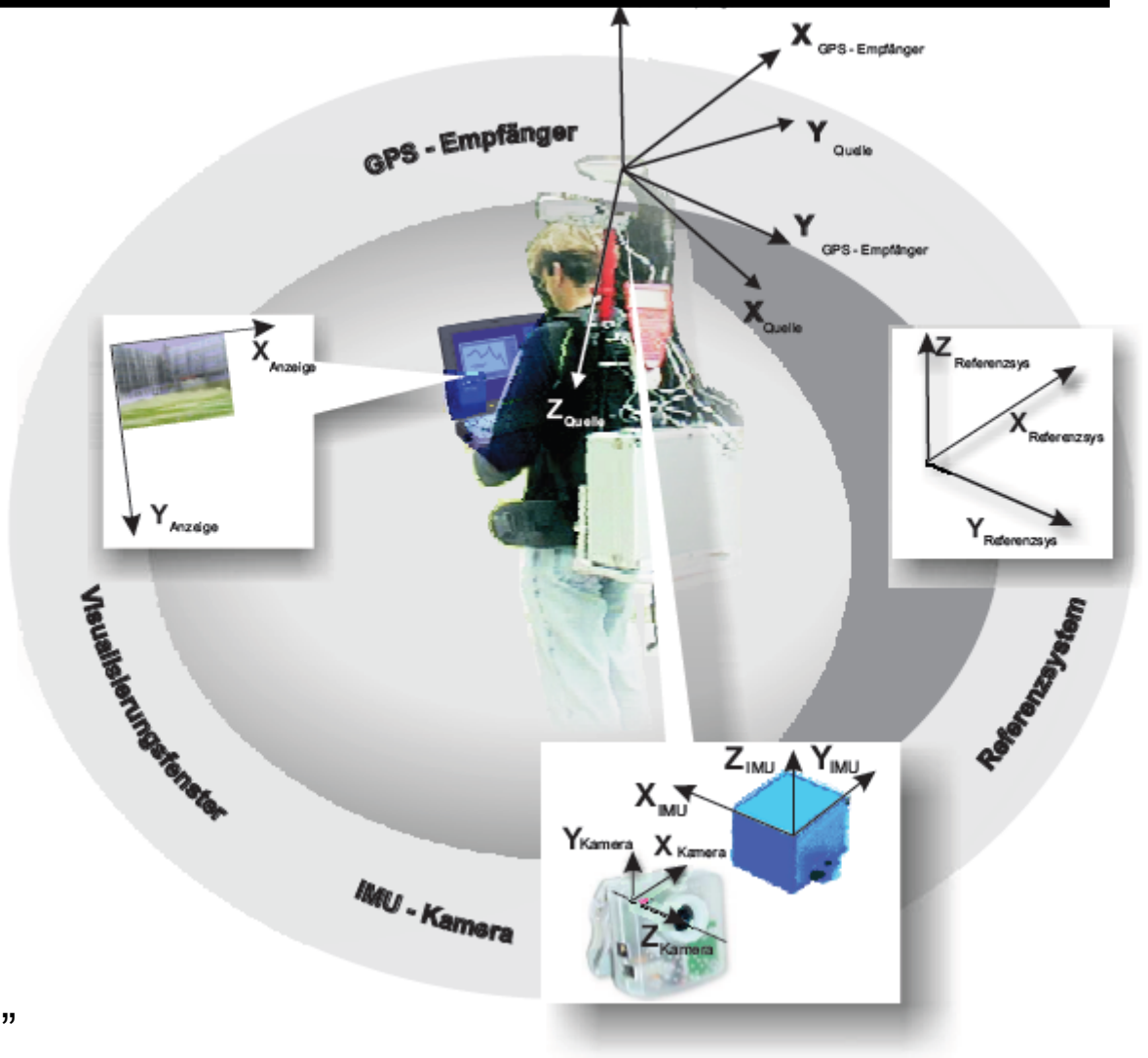
Zusammenführen der Messungen



(World) Quell-System:

- **Ursprung:** Zentrum GPS-Empfänger
- **Y** Ellipsoidtangente Richtung magnetisch Nord.
- **Z** Zenit.
- **X** Kreuzprodukt von **Y** und **Z**

Zusammenführen der Messungen



Geräte-Koordinaten

Ursprung: Screen unten links,
X horizontale Richtung, rechts,
Y vertikale Richtung, nach oben,
Z senkrecht auf Screen “Gerät aussen”

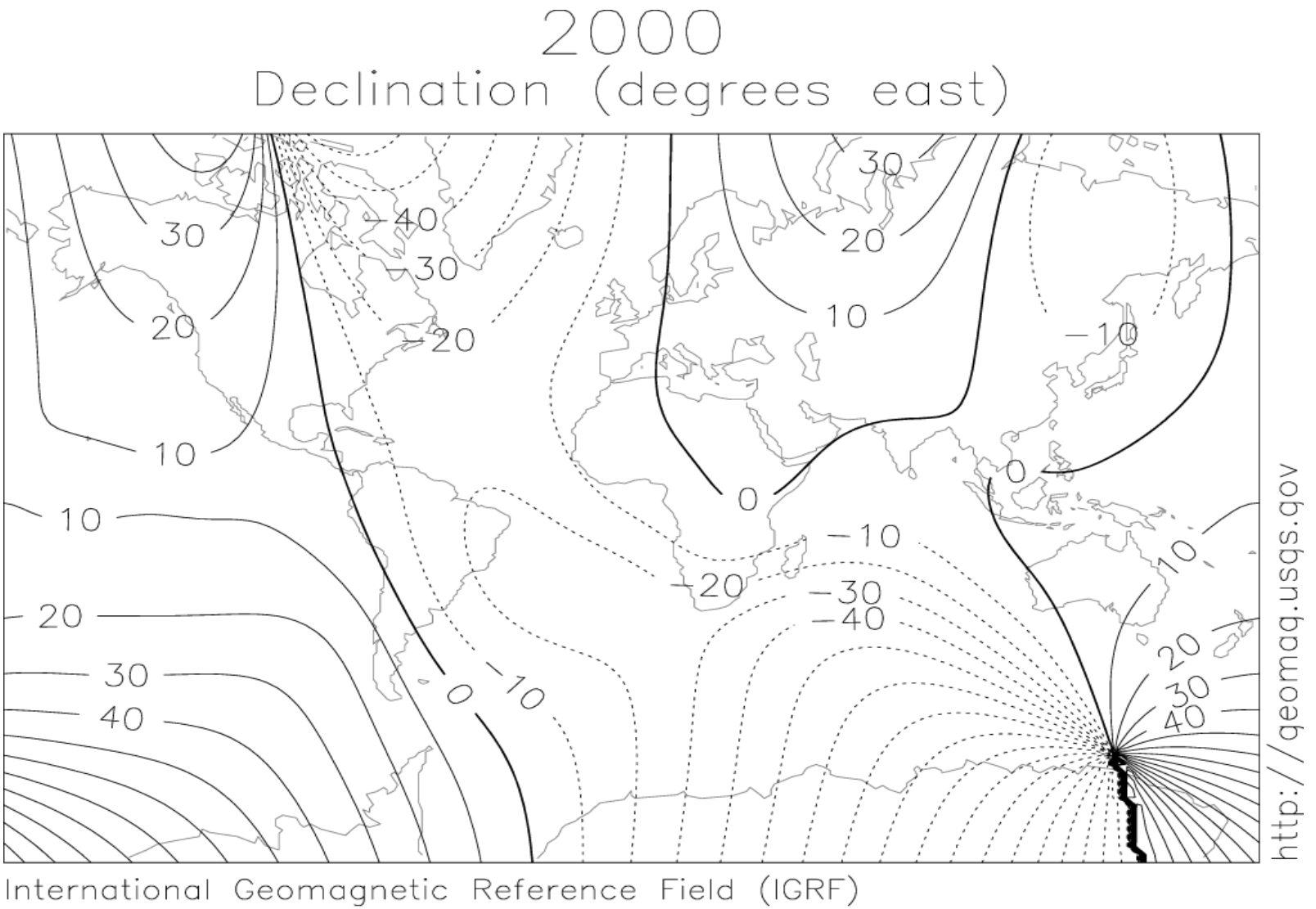
ClientSeitige API: SensorManager

Liefert Rotationsmatrix vom Quell-System in das Geräte-Koordinatensystem.
→ Problem: es fehlt die Kompassabweichung
→ Keine „Fusion“ von Kompass, Accelerometer und Schwere-Sensor
→ Keine Glättung

Beispiel Mixare

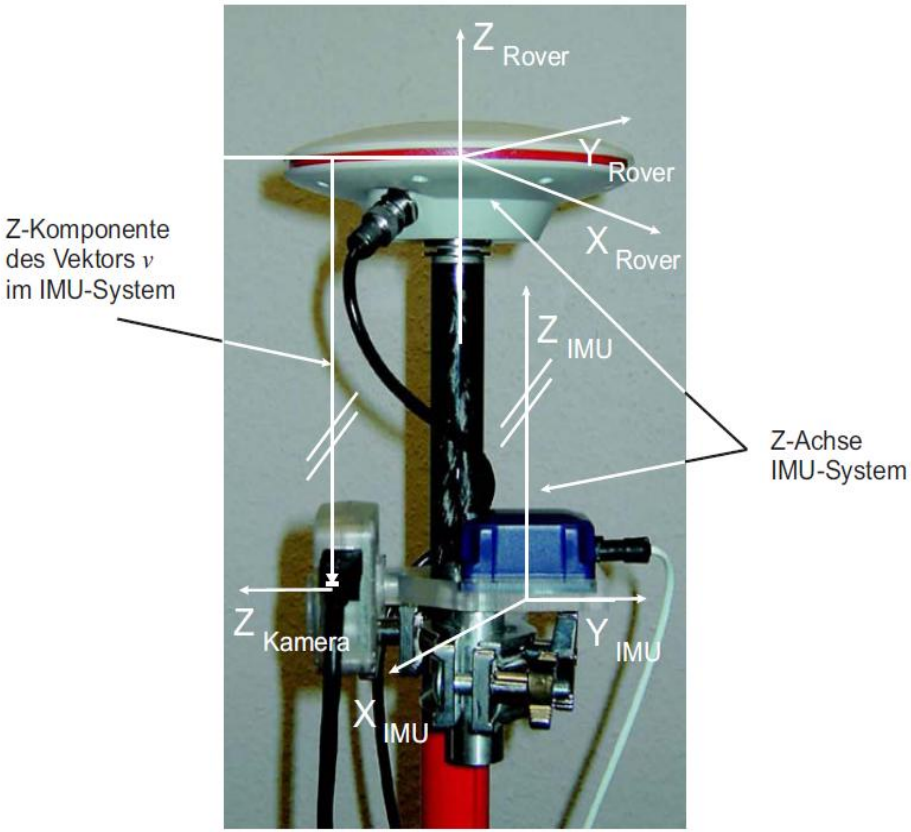
```
public void onSensorChanged(SensorEvent evt) {
    try {
        killOnError();

        if (evt.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
            grav[0] = evt.values[0];
            grav[1] = evt.values[1];
            grav[2] = evt.values[2];

            augScreen.postInvalidate();
        } else if (evt.sensor.getType() ==
   Sensor.TYPE_MAGNETIC_FIELD) {
            mag[0] = evt.values[0];
            mag[1] = evt.values[1];
            mag[2] = evt.values[2];

            augScreen.postInvalidate();
        }

        SensorManager.getRotationMatrix(RTmp, I, grav, mag);
```

Klasse GeomagneticField: Korrektur der Deklination



Zusammenführen der Messungen

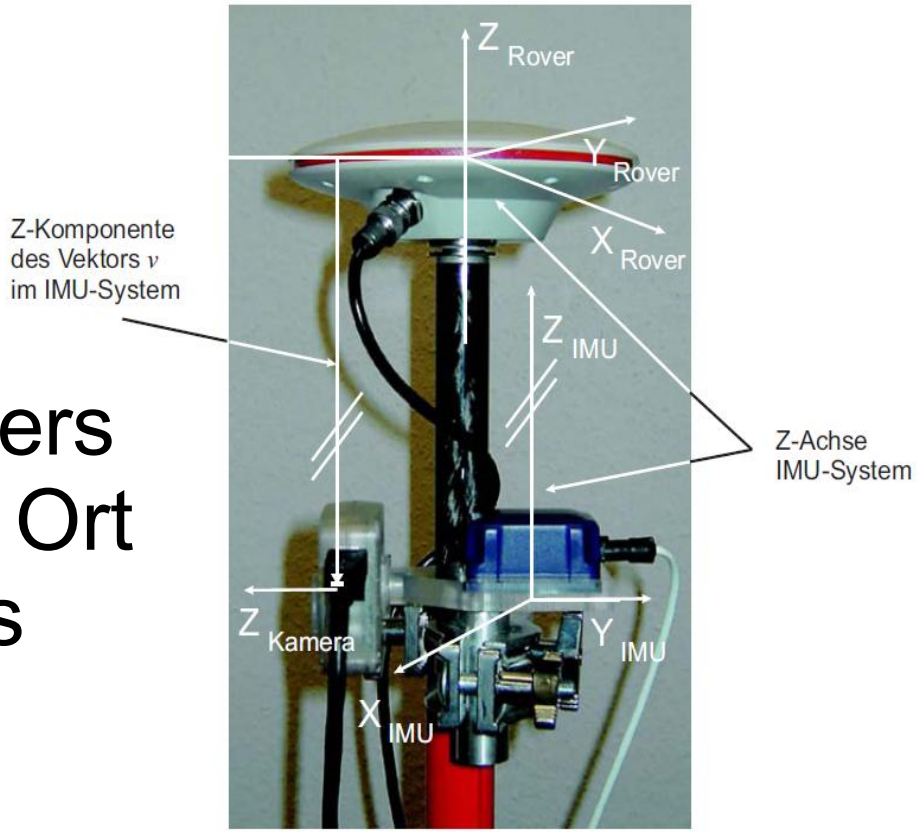
GPS-Empfänger-Quelle:
u.a. magnetische Deklination



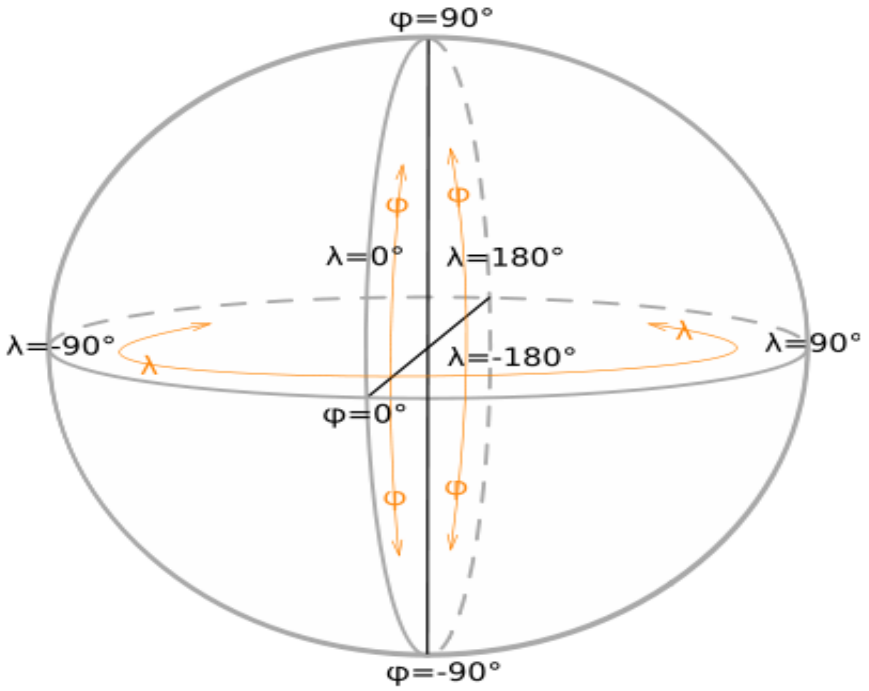
public GeomagneticField (float gdLatitudeDeg, float gdLongitudeDeg, float altitudeMeters, long timeMillis)

Clientseitige APIs - LocationManager

- Registrieren eines Listeners
- Liefert letzten bekannten Ort
- Definition eines Providers



LocationManager, Location



Zum Beispiel Mixare:
X Koordinate im Gerätekoordinatensystem
aus Abstand der Punkte auf gleicher geo.Länge

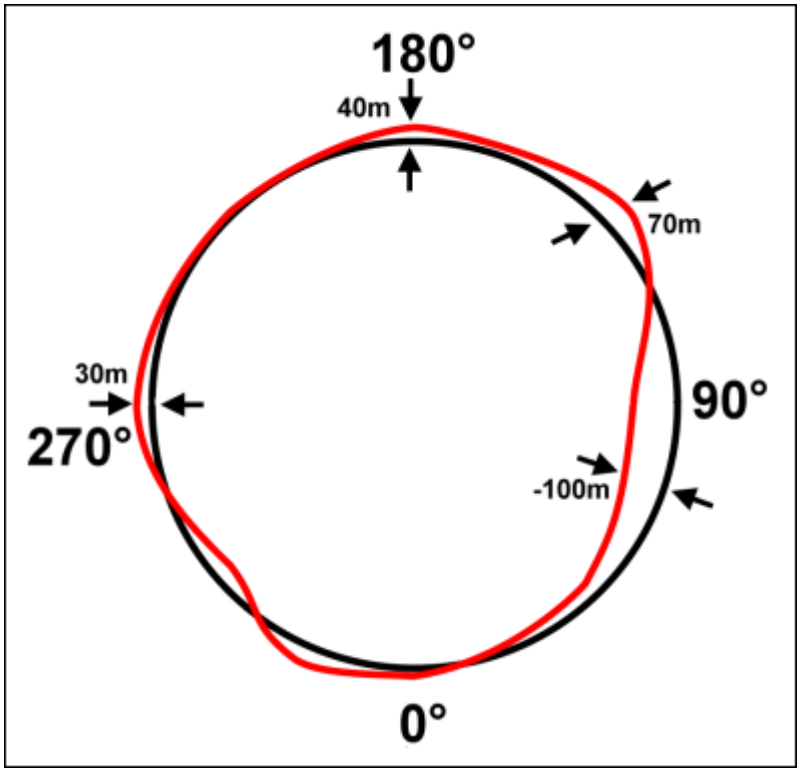
Fehler(Vernachlässigung der Erdkrümmung)
<<
Sensorungenauigkeit

```
Location.distanceBetween(org.getLatitude(),
org.getLongitude(), gp
.getLatitude(), org.getLongitude(), z);
```

Vereinfachung Location.getDistance()

Wappen	Deutschlandkarte
	

Basisdaten	
Bundesland:	Hessen
Regierungsbezirk:	Darmstadt
Landkreis:	Kreisfreie Stadt
Höhe:	144 m ü. NN
Fläche:	122,24 km²
Einwohner:	143.332 *(31. Dez. 2009)*[1]



Abweichung der Höhe
bezüglich Normal Null (NN) von Ellipse
→ Bei Geodaten, die sich auf NN beziehen
→ Hier kann es zu einem Höhenversatz kommen

ClientSeitige APIs: MarkerTracking

- Marker zur Bestimmung von Ort und Position des virtuellen Objektes
- Bereits kommerziell: Position in Innenräumen
- OpenSource: ARToolkit

Nachteil: Nutzung von vordefinierten Markern

Clientseitige APIs: OpenCV

Mehrere Portierungen verfügbar.
Zum Beispiel:

http://code.google.com/p/android-opencv/

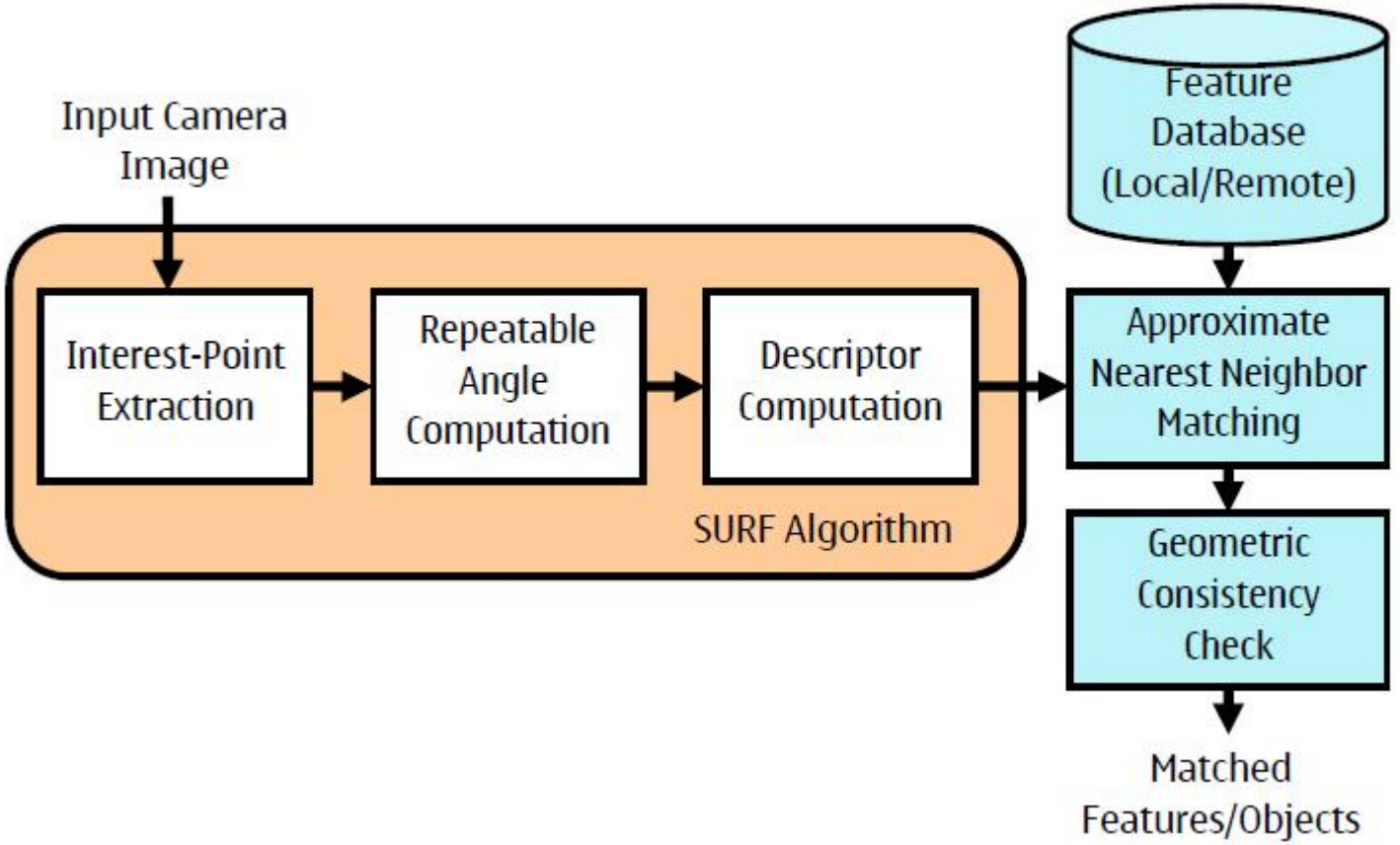
Basiert auf:

- crystax ndk r4
- swig
- New BSD License

Haar Classifier Generieren

Merkmale aus Trainingsdaten

Prinzip: Objekterkennung



Quelle: W.-C. Chen, et al 2007, Efficient Extraction of Robust Image Features on Mobile Devices

Fazit

- Android APIs nicht auskonsolidiert: Packages (Camera, Sensor, Location) und „Manager“-Klassen
- Kein Glättung für Sensoren (Kalman)
- Geoid-Undulation fehlt (vgl. Deklination)
- Sensor-Fusion fehlt

- Camera und OpenGL nicht „reibungsfrei“
- Szenengraph (vgl. Java3D) fehlt
- Bilderkennung nicht als API offen

Fazit

- Zugriff auf alles Sensoren über Java
- Wichtige Transformationen vorgefertigt
- Zugriff auf C++
- Hinreichend für Sensorgenauigkeit

Vielen Dank für die Aufmerksamkeit!