Entwicklung eines mobilen Karten-Frameworks für hochauflösende Grundrissdaten

Benny Platte

Hochschule Mittweida
Technikumplatz 17
09644 Mittweida, Germany,
platte@hs-mittweida.de

Zusammenfassung: Studien zu Indoor-Positionierung zeigen mehrheitlich retrospektive Analysen und Diagramme. Echtzeitfähigkeit wird als Schlagwort zwar verwendet, meint bei genauerer Analyse damit verklausuliert lediglich die Erfassung der Daten, nicht deren Auswertung. Eine funktionierende mobile Echtzeit-Visualisierung auf Grundrissdaten zeigt keine der untersuchten Studien, bei allen werden Grundrisse nur zur retrospektiven Darstellung verwendet.
Diese Arbeit realisiert eine Echtzeit-Visualisierung von Indoor-Positionsdaten auf einem hochauflösenden Grundriss. Dafür werden die technischen und praktischen Problemstellungen aufgezeigt und Anforderungen abgeleitet. Hierfür werden einzelne Lösungen vorgestellt, daraus eine Prozesskette entwickelt und eine nachweislich tatsächlich in der Praxis funktionierende mobile Applikation mit einer Echtzeit-Visualisierung vorgestellt. Dazu wird neues Kartenmaterial generiert und die Funktionalität der Mobilgeräte-Programmierschnittstelle dahingehend erweitert, dass eine nahtlose Integration in die Kartierungsdarstellung des Mobilgeräts mit Erweiterung bis in den Intraraumbereich erfolgt.

Schlagwörter: Indoor-Localization, Mapping, Cartography, Realtime Visualization

1 Einleitung

Je nach System liegen ermittelte Positionen in Innenräumen in unterschiedlichen Parametern vor, z.B. : „Regal 3“, „in Reichweite des Routers im Wohnzimmer“ oder in proprietären Lokalkoordinaten. Bei ausschließlicher Verwendung in proprietären Systemen wie beispielsweise einer Visualisierung von Hotelzimmern besteht kein Handlungsbedarf. Bei weitergehenden Einbindungen, z.B. für kartenbasierte Integrationen oder Orientierung in Gebäudekomplexen müssen Positionen von Entitäten in ein Raumkonzept eingebettet werden. Insbesondere für die Dienstintegration in Mobilgeräte bieten die herstellereigenen Programmierschnittstellen wie OpenStreetMap [Ram22], Google Maps [Goo22] oder Mapkit von Apple [App22b] einen auf den ersten Blick hindernisarmen Zugang zu Mappingdiensten.

Diese Dienste sind an Voraussetzungen gebunden wie die Bereitstellung von Positionsdaten in einheitlichen Koordinaten. Des Weiteren unterliegen die Frameworks sämtlicher

Hersteller ähnlichen Restriktionen in Bezug auf die angebotene Genauigkeit und Auflösung. Die mit den herstellereigenen APIs erreichbare Auflösung beträgt im Mittel reale $0{,}3\,\text{m}/\text{px}$, was – bezogen auf die Displaybreite – der Abbildung einer Strecke von etwa $100\,\text{m}$ entspricht. Bei dieser Auflösung sind intraraumbezogene Informationen nicht praktikabel darstellbar.

In dieser Arbeit werden Lösungen vorgestellt, um Entitäten innerhalb von Räumen insbesondere auf Mobilgeräten praxisverwendbar darzustellen. Dazu werden Lösungen für die sich aus den Restriktionen ergebenden 3 Probleme vorgestellt:

- Anpassung der internen Datenhaltung an die erhöhten Genauigkeitsforderungen und deren Transformation in die API des Herstellers
- Erweiterung der Darstellbarkeit in höherer Auflösung
- Erzeugung entsprechenden Kartenmaterials mit Koordinatentransformation

Dies reicht von proprietären Integrationen („markiere Regal 3 im Warenwirtschaftssystem“) bis hin zu Transformationen in ein globales Koordinatensystem. Derartig transformierte Positionen können – ein geeignetes Koordinatensystem vorausgesetzt – von jeder Software in einer Kartendarstellung visualisiert werden. Die Zahl solcher Anbieter ist überschaubar, bei Einsatz auf Mobilgeräten verbleiben im Wesenlichen nur Alphabet, Apple und Microsoft.

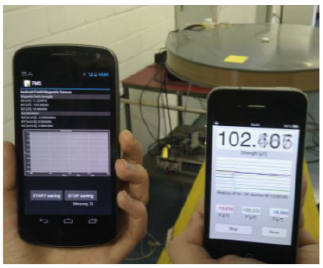
2 Verwandte Arbeiten

Die berichteten Genauigkeiten für Indoor-Positionsdaten scheinen zunächst mehr als ausreichend. Arumugam et al. berichten von $20{,}3\,\text{cm}$, Shahjalal et al. von $2-8\,\text{cm}$ und Zheng et al. von $5\,\text{cm}$ [Aru+11; Sha+18; Zhe+20]. Eine Größenordnung darunter ($1\,\text{mm}$) erreichen Huang et al. [Hua+15]. Die reine Auflösung soll hier nicht diskutiert werden, sie steht zudem im Zusammenhang mit dem jeweiligen Erfassungsbereich. So erreichen Arumugam et al. zwar $20\,\text{cm}$, dies aber in einem begrenzten Bereich von $25\,\text{m}$. Die $1\,\text{mm}$ Auflösung von Huang et al. bezieht sich auf einen Bereich von nur $150\,\text{mm}$.

All diese Arbeiten haben gemein, dass die Daten in einem *proprietären System* vorliegen. Die Visualisierung der Positionsdaten erfolgt dabei

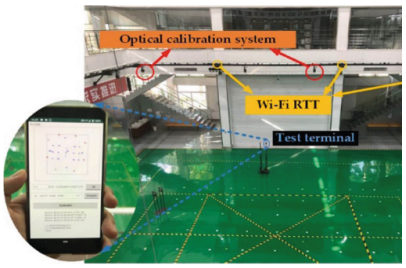
- überhaupt nicht [Zhe+20; Sha+18; Yan+15; Rah+17].
- nur in Form von retrospektiv aus den Daten erzeugten Diagrammen [Aka+18, S. 647; Cer+20] oder groben Heatmaps [Cha+21, S. 9; LB21, S. 3].
- als Trajektoren ohne äußeres Koordinatensystem [Lu+17, S. 147; Liu+20, S. 1, 4].
- retrospektiv in Grundrissen über generierte Punktwolken [Grz+20; Hes+16, S. 1275; Mao+20, S. 2].
- retrospektiv in echten Grundrissen mit überlagerten Lokalisierungen [PLP21, S. 2; KGA17, S. 20].

Echtzeit-Positionsangabe ganz ohne Mapping: Sprachausgabe.



Echtzeit-Positionsangabe in kleinem Umkreis. Funktionierende App.

Smartphones verwendet, aber nur zur „realtime" Aufzeichnung.



prototypische Positionsdarstellung als Rechteck mit Punkten in abgegrenztem Bereich.

Abbildung 1: Smartphone-Verwendung anderer Autoren. Von links nach rechts: Al-Khalifa und Al-Razgan [AA16], Blankenbach, Norrdine und Hellmers [BNH12], Akal et al. [Aka+18], Huang et al. [Hua+21]

Viele Autoren bezeichnen die Positionierung zwar als „real time", stellen diese aber nur als Wording oder in retrospektiven Diagrammen oder Grundrissen dar. Selbst Studien, die im Text von „real time" oder „Echtzeit" in Verbindung mit „Smartphones" sprechen, stellen keine aussagekräftigen Abbildungen *tatsächlicher Funktionalität* zur Verfügung [LDW18; Mor+21]. Abbildungen in Form von Schemazeichnungen werden zwar häufig verwendet, *in praxi* funktionierende und während des Betriebes dargestellte Smartphone-Anwendungen sind jedoch selten anzutreffen; nur Al-Khalifa und Al-Razgan zeigen eine Anwendung zur Positionsbestimmung für Menschen mit Seheinschränkungen [AA16] und Huang et al. eine in Aktion befindliche Smartphone-Anwendung für Wifi-basierte Positionsbestimmung [Hua+21].

3 Problemstellung

Sollen bereitstehende Indoor-Positionsdaten, welche in einem wie auch immer gearteten proprietären Bezugssystem vorliegen, transparent in eine mobile Anwendung eingebunden werden, ergeben sich drei Problemfelder:

1. die Begrenzung der Zoombarkeit in den Kartierungsanwendungen und Programmierschnittstellen *aller* Hersteller auf Werte, die eine Indoor-Positionierung nahezu unmöglich machen.
2. eine prinzipbedingte Einschränkung durch die verwendete Repräsentation der Zahlen im mobilen Rechenwerk und damit der Genauigkeit des verwendeten Zahlensystems.
3. Fehlendes Kartenmaterial in entsprechender Auflösung.

Diese Punkte werden im Folgenden beleuchtet.

3.1 Auflösung/Zoombarkeit

Sämtliche Hersteller der Programmierschnittstellen von Karten-APIs begrenzen die Darstellbarkeit bzw. Anzeige auf Werte, die für eine zimmergenaue Lokalisierung untauglich sind; Eine Übersicht der marktbeherrschenden Hersteller zeigt Tab. 1. Für die interne Berechnung verwenden alle das Konzept der *Zoomstufen*. Die Zoomstufe gibt den Detailgrad an, der auf der Karte angezeigt werden soll; größere ganze Zahlen führen zu einem höheren Detailgrad [Tom22b].

Tabelle 1: Real verfügbare Auflösung (d.h. inklusive der „leeren Vergrößerung“ [Paw90, S. 209]) unterschiedlicher Kartierungsanbieter auf einem Bildschirm

API-Anbieter	Auflösung in $^{\text{Pixel}}/_{\text{Meter}}$		Reference	Marktanteil bei mobilem Betriebssystem [Kun17]
	Rasterkarten	Vektorkarten		
Google	6,3..19	21,4	[Goo22]	71,3 % (Android)
Apple	5,6	8,8	[App22c]	20,1 % (Apple iOS)
Microsoft	5,6	10,8	[Mic22]	5,7 % (Windows Phone)
Tomtom	7,6	12,2	[Tom22a]	k.A.

Die herstellerseitige Begrenzung der Karten-APIs bei Apples *Mapkit* sieht eine Sperrung aller *z-Level* ab „19“ vor [App22b]. „18“ war damit die letzte auf einem Bildschirm anzeigbare Zoomstufe. Bei einer Auflösung von $2\,^{\text{px}}/_{\text{m}}$ konnte damit auf dem Bildschirm des Testgeräts bei maximalem mnanuellen Zoom ein Bereich von $223\,\text{m}$ Breite angezeigt werden (gelbe Markierung in Tab. 1). Dieser Wert wurde nur möglich durch ein herstellerseitiges verlustbehaftetes Vergrößern der Kartenkacheln. Die Hardwareauflösung des Gerätebildschirms betrug in der Breite $750\,\text{px}$, was bei der letzten Zoomstufe „18“ einer Kachelauflösung von $2\,^{\text{px}}/_{\text{m}}$ entspricht. Der Bildschirm würde dann einen Kartenbereich von $448\,\text{m}$ zeigen ($700\,\text{m}$ bei 1170 Hardware-Pixeln eines iPhone 13). Um dem Benutzer einen höheren Zoom vorzugaukeln, erfolgt intern eine Verdreifachung der Auflösung und eine Zusammenfassung von jeweils 9 „Hardware-Pixeln“ zu 1 „logischen Pixel“. Dies lässt den Nutzer mittels „Pinch“-Geste zwar weiter vergrößern, aber der Informationsgehalt steigt nicht weiter; Es handelt sich um eine „leere Vergrößerung“ [Paw90, S. 209]. Dieser Eigenschaft bedienen sich bereits seit Jahrzehnten Mikroskophersteller, um in ihren Datenblättern Vergrößerungen angeben zu können, die gleichauf mit um Größenordnungen teureren Geräten liegen. Eine solche Vergrößerung erzeugt jedoch „keine weitere Auflösung“ und wird „nicht mehr als förderliche Vergrößerung, sondern als leere Vergrößerung bezeichnet“ [KRW15, S. 7]. Der tatsächliche Informationsgehalt auf einem Bildschirm ist für eine Intraraumpositionierung ungeeignet.

3.2 Genauigkeit des Zahlensystems

Sämtliche Anbieter von Mapping verwenden das WGS84-System in der Spezifikation „EPSG:3857“ als Modellierung des Erdkörpers mit einer Koordinatenangabe in Längen- und Breitengrad [Kam22]. Die Programmierschnittstelle (API = Application Programming Interface) aller Anbieter stellt diese im Datentyp „Double“ nach IEEE-Standard 754 in der letzten Revision von 2019 dar („binary64“ in [IEEE19, S. 18]). Damit sind bei einer Speichertiefe von 64 Bit und einer Mantisse von 52 Bit ca. 15 signifikante Ziffern speicher- bzw. darstellbar. Werden lokal erzeugte Positionsdaten im Millimeterbereich in das globale WGS84-System konvertiert, finden dessen Änderungen im Grenzbereich des IEEE-754-Formats statt: Fehler kumulieren auf und die Position wird entsprechend ungenau.

3.3 Kartenmaterial, Koordinatentransformation

Für die Einbindung in ein globales Koordinatensystem sind – unabhängig von den Genauigkeitsanforderungen aus dem vorhergehenden Abschnitt – geeignete Karten notwendig, die nicht nur in entsprechenden Auflösungen zur Verfügung stehen müssen, sondern darüber hinaus in das Koordinatensystem des Kartenherstellers transformiert werden müssen. Dies beinhaltet ein „Einrasten“ an einem Referenzpunkt und eine Rotation des Koordinatensystems. Die Kartenhersteller verwenden hierfür verschiedene Koordinatensysteme, die Transformationen sind demnach jeweils anzupassen.

4 Methoden

4.1 Genauigkeit des Zahlensystems

Die Anforderungen der Genauigkeit stehen hier an erster Position, da sie die Grundlage sämtlicher Berechnungen bilden. Der Datentyp „binary64“ aus dem IEEE-754-Format stellt prinzipiell genügend Genauigkeit zur Verfügung, um im Submillimeterbereich Positionsparameter anzugeben, nicht jedoch, im Grenzbereich vielfältige fortgesetzte Additionen auszuführen. Um diese Fehlerakkumulation zu vermeiden, wurden zwei prinzipielle Wege untersucht: Zum einen ist es möglich, die Klassen umzustellen auf Datentypen mit höherer Auflösung, zum anderen werden sämtliche Berechnungen und Additionsketten im lokalen Koordinatensystem ausgeführt und erst zum Visualisierungszeitpunkt in den Datentyp der API konvertiert.

Die Methode der Umstellung auf Datentypen höherer Auflösung bietet den Vorteil, im Moment der Datengewinnung sofort zu konvertieren und danach nur noch im Zielkoordinatensystem zu rechnen. Dem steht als Nachteil gegenüber, die notwendigen Änderungen

der Datentypen einige Ebenen unterhalb der API vornehmen zu müssen, was eine Änderung von Programmcode voraussetzt, welcher vom Hersteller bereits getestet wurde und immer dann angepasst werden muss, sobald vom Hersteller eine Änderung publiziert wird. Da die Änderungen in einem Bereich vorgenommen würden, der nicht von offiziell dokumentierten Schnittstellen umfasst ist, würde dieses Vorgehen zu einem erhöhten Pflegeaufwand führen. Überdies würden Apps, die solche Änderungen in einem nicht vertrauenswürdigen Bereich („untrusted Code“) vornehmen, bei regelmäßigen Überprüfungen aus dem entsprechenden Appstore entfernt. Aus diesen Gründen wurde dieses Vorgehen verworfen.

Nach dem Verwerfen der Umstellung der Datentypen wurde das Verfahren der kompletten Berechnung und Kettenaddition im Lokalkoordinatensystem mit „später Transformation“ umgesetzt: Erst im Moment der Kartendarstellung wird das entsprechende Datum aus der Liste der lokalen Koordinaten in das Projektionssystem „EPSG:3857“ transformiert. Weitere Berechnungen der Position finden immer im lokalen System statt, welches dadurch in sich konsistent bleibt. Eine neue relevante Position wird immer aus den Lokalkoordinaten berechnet. Die Kartenkoordinaten dienen nur dem Benutzerinterface und spielen bei der Positionsberechnung gemäß dem Entwurfsprinzip „Separation of Concerns“ (SoC) keine Rolle [Pla+20, S. 359; Wik19].

4.2 Auflösung/Zoombarkeit

Die objektorientierte Repräsentation der Daten und Funktionen innerhalb der Entwicklungsumgebungen der Hersteller ermöglicht, Funktionsbereiche im Nachhinein mit weiterer Funktionalität auszustatten. Technisch wird dies über Methoden der Vererbung („Inheritance“ [LRS21, S. 242; Mus21, S. 314]) oder Erweiterungen („Extensions“ [App22a] realisiert. Für die Bereitstellung eigener Karten bieten die Hersteller der Kartierungs-APIs für Smartphones ähnliche Schnittstellen an. Im Fall von *Apple MapKit* wurde eine Schnittstelle verwendet, die es ermöglicht, eigene Karten statt der bekannten Vektor- oder Satellitenansicht bereitzustellen [App22b]. Eine Erweiterung dieser Standardklasse ermöglicht statt der bisher verwendeten Koordinaten *longitude* und *latitude* sowie *z-Level* weitere, eigene Eigenschaften zu definieren. Die Standard-Implementierung sperrt *z-Level* ab einer bestimmten – vom Hersteller festgelegten – Vergrößerungsstufe. Dieser Wert wird an die Ursprungsklasse durchgereicht und steuert die Anzeige während eines Zoomvorganges. Unerlaubte *z-Level* werden ignoriert. Eigene Kartierungsklassen können einen höheren Wert an das System durchreichen, wenn sie die Visualisierung dafür in einem eigenen Renderer selbst implementieren.

Über diesen Umweg wird es möglich, Vergrößerungswerte der Oberfläche jenseits der herstellerseitigen Beschränkungen mit eigenen Werten zu überschreiben. Die Integration wurde mit einigem Aufwand mittels der Implementierung eines eigenen Renderers

`hsmw_MapViewController` nahtlos gestaltet: bis zur Herstellergrenze kann die Karte des Herstellers angezeigt werden, aber statt ein weiteres Vergrößern zu verhindern, wird eine neue, benutzerdefinierte Karte sichtbar.

Tabelle 2: Zoomlevel für die Bereitstellung von Kartenkacheln mit Auswirkungen auf die Bildschirmanzeige sowie der prinzipiell notwendigen Genauigkeit und dem maximalen offiziell verfügbaren Detailgrad (gelb).

	Zoom-level	Kartenkacheln			Bildschirmbezogene Angaben				notwendige Genauigkeit	
		Anzahl Strecken-abschnitte	Anzahl der Kartenkacheln („Tile Count“)	Längengrad Winkelbereich [°]	Kachellänge am Äquator [m]	Auflösung: *wahre* Pixel $\left[\frac{m}{px}\right]$	zoombarer Anzeigeumfang des Bildschirms (Breite) [m]	[m]	Längengrad: benötigter Inkrement-umfang	Längengrad-Inkrement bei 1 Pixel [°]
verfügbare Kartenelemente des Herstellers	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	14	16 384	268 435 456	0,021 972 7	2446	0,105	3582	7165	4 194 304	$8{,}58{\cdot}10^{-5}$
	15	32 768	1 073 741 824	0,010 986 3	1223	0,209	1791	3582	8 388 608	$4{,}29{\cdot}10^{-5}$
	16	65 536	4 294 967 296	0,005 493 2	611,5	0,419	895	1791	16 777 216	$2{,}15{\cdot}10^{-5}$
	17	131 072	17 179 869 184	0,002 746 6	305,7	0,837	447,8	895,7	33 554 432	$1{,}07{\cdot}10^{-5}$
	18	262 144	68 719 476 736	0,001 373 3	152,9	1,67	223,9	447,8	67 108 864	$5{,}36{\cdot}10^{-6}$
neu generierte Elemente	**19**	524 288	274 877 906 944	0,000 686 6	76,44	3,35	111,9	223,9	134 217 728	$2{,}68{\cdot}10^{-6}$
	20	1 048 576	1 099 511 627 776	0,000 343 3	38,22	6,70	55,98	111,9	268 435 456	$1{,}34{\cdot}10^{-6}$
	21	2 097 152	4 398 046 511 104	0,000 171 7	19,11	13,4	27,99	55,98	536 870 912	$6{,}71{\cdot}10^{-7}$
	22	4 194 304	17 592 186 044 416	0,000 085 8	9,555	26,8	13,99	27,99	1 073 741 824	$3{,}35{\cdot}10^{-7}$
	23	8 388 608	70 368 744 177 664	0,000 042 9	4,777	53,6	6,998	13,99	2 147 483 648	$1{,}68{\cdot}10^{-7}$
	24	16 777 216	281 474 976 710 656	0,000 021 5	2,389	107	3,499	6,998	4 294 967 296	$8{,}38{\cdot}10^{-8}$
mögliche Erweiterung	**25**	33 554 432	1 125 899 906 842 620	0,000 010 7	1,194	214	1,750	3,499	8 589 934 592	$4{,}19{\cdot}10^{-8}$
	26	67 108 864	4 503 599 627 370 500	0,000 005 4	0,5972	429	0,875	1,750	17 179 869 184	$2{,}10{\cdot}10^{-8}$
	27	134 217 728	18 014 398 509 482 000	0,000 002 7	0,2986	857	0,437	0,875	34 359 738 368	$1{,}05{\cdot}10^{-8}$
	28	268 435 456	72 057 594 037 927 900	0,000 001 3	0,1493	1710	0,219	0,437	68 719 476 736	$5{,}24{\cdot}10^{-9}$

4.3 Kartenmaterial, Koordinatentransformation

Nach Implementierung der erweiterten Klassen stand war es möglich, über den *z-level* den Zoomfaktor prinzipiell anzuheben. Eine tatsächliche Darstellung auf dem Display des Mobilgerätes erfordert zusätzliches Kartenmaterial in geeigneter Form.

Ausgangslage für Ersteres war ein im CAD-Format vorliegender Grundriss des Gebäudes. Dieser wurde in ein pixelbasiertes Format umgewandelt mit einer Auflösung von 400 px/m. Für den betreffenden Bereich wurden Open-Street-Map-Kartenkacheln in der höchsten verfügbaren Auflösung „18“ im Projektionssystem „Web Mercator EPSG:3857“ geladen [Ram22; EPSG22] und aneinandergefügt. Diese Kacheln weisen eine standardi-

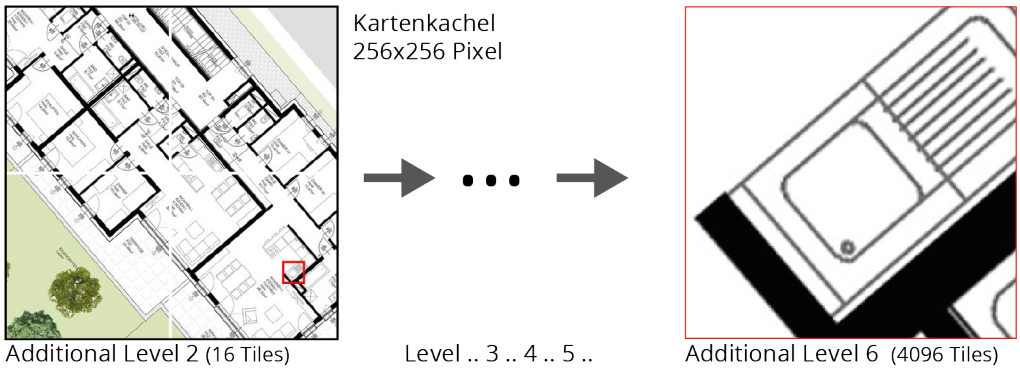


Abbildung 2: Kartenkacheln: Vergleich und bildliche Darstellung der zusätzlich erzeugten Zoomstufen 20 und 24 aus Tab. 2.

sierte Größe von jeweils 256×256 Pixel auf. Diese Auflösung wurde ver-64-facht auf eine Kachelauflösung von 16384×16384 Pixel (Abb. 2). Die Skalierung musste keinerlei Qualitätsansprüchen genügen, eine Interpolation ist nicht notwendig; skaliert wurde mittels einfachem Vervielfältigen der Pixel. Diese Kacheln dienten als Underlay für den hochauflösenden gerasterten Grundriss.

Anschließend erfolgte die Überlagerung mit dem hochauflösenden Raster-Grundriss auf Basis des festgelegten Referenzpunktes. Dieser wurde im Grundriss exakt vermessen und einer Koordinate im Projektionssystem des Kartenherstellers zugeordnet. Die Rotation gegen Nord wurde ebenfalls dem Grundriss entnommen und dieser entsprechend rotiert.

Nach der Transformation des Referenzpunktes des Grundrisses in das pixelbasierte Koordinatensystem der Kartenkacheln wurde das Underlay mit dem Grundriss überlagert und als einzelne hochauflösende Rastergrafik exportiert.

Aus dem Grundriss wurde mit dem Ziel, später Daten zu sparen, eine weitere hochauflösende Rastergrafik exportiert: Die Grenzen des Grundrisses wurden pixelscharf maskiert und diese Maske einheitlich gefärbt. Alle für den Grundriss unwichtigen Teile wurden auf diese Weise maskiert.

Die Erzeugung der Kartenkacheln in den Zoomstufen $19 \ldots 24$ wurde mittels einer selbst entwickelten Software `hsmw_maptiler` durchgeführt. Dieser werden zwei sehr hoch aufgelöste Kartenkacheln („Tiles“) übergeben: das „Ursprungs“-Tile mit dem transformierten Grundriss und das „Masken“-Tile. Beide haben - ein bestimmtes zugrunde liegendes Zoomlevel, im vorliegenden Fall 18 - eine auf alle in x-Richtung vorhergehenden Kacheln bezogene x-Koordinate bzw. einen nullbasierten Index (bei Zoomlevel 18 im Bereich von $2^0 = 1$ bis $2^{18} = 262144$) - eine y-Koordinate bzw. einen nullbasierten Index in y-Richtung (bei Zoomlevel 18 ebenfalls im Bereich von $2^0 = 1$ bis $2^{18} = 262144$)

Diese Werte müssen zwingend angegeben werden; Aus diesen werden die neuen Koordinaten aller weiteren Zoomstufen gebildet. Ein Parameter ist „zoomstep“. DIeser gibt an, wie viele Zoomschritte über dem Ursprungs-Zoomlevel die Tiles generiert werden sollen. Beispieldurchgang für $zoomstep = 2$:

- Entlang einer Achse sollen $2^{zoomstep}$ Tiles generiert werden, also wird die Kachel horizontal und vertikal in jeweils 4 Stücke geschnitten, insgesamt also $2^{zoomstep} \cdot 2^{zoomstep} = 16$ Tiles aus dem Ursprungsbild generiert.
- Das Tile links oben hat nun die Koordinaten $x = 1048576$. Erklärung: Die Tile-Koordinaten sind immer absolut. Wird ein Zoomlevel um eine Stufe erhöht, befinden sich damit auch doppelt so viele Tiles vor dem aktuellen. Da im Beispiel 2 Zoomstufen hochgesprungen wird, befinden sich nun $(262144 \cdot 4 - 1)$ Tiles vor dem übergebenen Tile (bezogen auf die gewünschte Zoomstufe). Damit beträgt die Koordinate des ersten neuen Tiles das $2^{zoomstep}$-fache des Ursprungs-Tiles.
- Bei jeder berechneten Neu-Koordinate prüft `hsmw_maptiler` den entsprechenden Bereich des Masken-Tiles. MIttels eines „Area Coverage Check“ wird der Betrag der Überlappung der neuen Kachel mit der Maske geprüft: bei vollständiger Bedeckung wird keine neue Kachel berechnet, da *an dieser Stelle* kein zusätzlicher Informationsgehalt aufgrund des Fehlens von hochauflösenden Grundrissdaten besteht.
- Das generierte Tile wird in einer festen Ordnerstruktur abgelegt: `zoomlevel/xindex/yindex.png`

5 Ergebnisse

Im vorliegenden Mobilgerät (iPhone 7 [App21] wurde ein vorerst unsichtbares Overlay über die herstellereigene Karte implementiert. Nach Überschreiten der vormaligen Zoomgrenze wird das Overlay sichtbar. Im vorliegenden Fall wurden – startend bei *z-level* 19, sechs zusätzliche Ebenen implementiert: Die Herstellergrenze von $2\,\text{px}/\text{m}$ wurde bis $z_{level} = 24$ erhöht auf eine Darstellbarkeit von $107\,\text{px}/\text{m}$. Pro Ebene erfolgt theoretisch eine Verdopplung der Pixel pro Längeneinheit. Mit der Bereitstellung von 6 weiteren Ebenen konnte die Auflösung um den Faktor 64 erhöht werden: Die Fläche der letzten verfügbaren Kartenkachel bei Zoomlevel 18 belegten in $n(z_{level} = 24) = 4096$ neue Kartenkacheln.

Die Software `hsmw_maptiler` erzeugte insgesamt 1885 Kacheln in 6 weiteren Zoomstufen (Tab. 3).

Der implementierte Renderer übernimmt ab Zoomstufe 18 die Kontrolle über die Darstellung der Karte. Abb. 3 zeigt Screenshots der erstellten Gesamtapplikation. Die generierten Kartenkacheln werden aus dem lokalen Speicher des Mobilgeräts geladen und angezeigt. In Erweiterung des Renderers wurde eine Repräsentation im URL-Format implementiert. Hierdurch ist es möglich, die Quelle für die Kacheln dynamisch festzulegen. Der

Tabelle 3: von `hsmw_maptiler` erzeugte Kartenkacheln

zoomlevel:	19	20	21	22	23	24
theoretisch zu erzeugende Tiles:	4	16	64	256	1024	4096
tatsächlich erzeugte Tiles:	3	8	29	94	355	1396

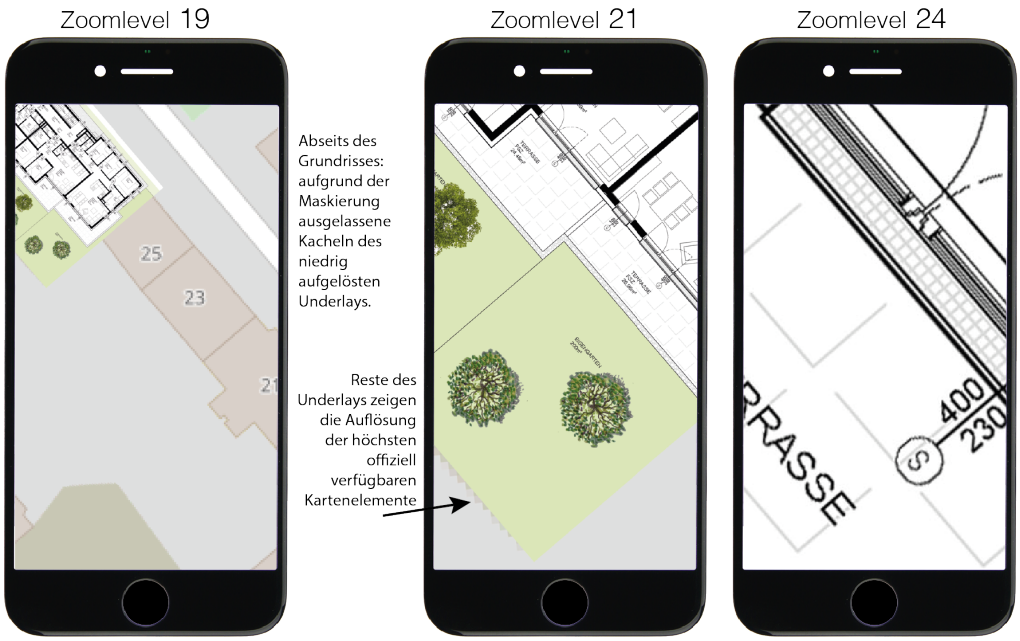


Abbildung 3: Display-Screenshots der Zoomstufen. Links der maximal erreichbare Zoom mit herstellereigener Karte, bereits überlagert mit Grundriss. Rechts daneben mit entwickeltem Framework erreichte (Zwischen-) Zoomstufe 21 und maximale Zoomstufe 24.

Kartenbestand aktualisiert sich bis Zoomlevel 18 automatisch, da die nicht im Cache befindlichen Kacheln dynamisch über eine Webverbindung nachgeladen werden können. Des Weiteren werden Aufrufe auf nicht vorhandene (maskierte) Kacheln umgeleitet auf vorher festgelegte oder dynamisch nachzuladende Kacheln, um den Nutzerinnen zusätzliche Informationen bereitzustellen.

6 Diskussion

Die Applikation läuft autark unter dem Betriebssystem iOS. Ein Benutzer sieht die Karte, kann scrollen und zoomen wie immer. Befindet sich der Nutzer in einem Bereich, in welchem ein Grundriss verfügbar ist, kann er oder sie nahtlos hineinzoomen. Die üblichen

Beschränkungen des Zoombereichs sind innerhalb der Applikation aufgehoben; Der Nutzer bzw. die Nutzerin bemerkt keinen Unterschied: Zoomen ist möglich bis Zoomlevel 24, was in der vorliegenden Kartenprojektion „EPSG:3857“ einer Auflösung von $107\,\mathrm{px/m}$ entspricht. Das Mobilgerät zeigt dabei einen Bereich von $3{,}5\,\mathrm{m}$ formatfüllend bezogen auf die Bildschirmbreite anstatt vorher $223\,\mathrm{m}$ (Abb. 3). Verglichen mit der Standard-Kartenapp des Herstellers – im vorliegenden Fall Apple – kann eine „Auflösungsmauer“ durchstoßen und gefühlt beliebig weit in ein Gebäude hinein gezoomt werden.

Die theoretisch notwendigen 5460 Kacheln konnten durch Maskierungsmaßnahmen auf 1885 reduziert werden. Dadurch konnte der Speicherbedarf um $65\,\%$ gesenkt werden. Die 1885 Kacheln belegen $50\,\mathrm{MiB}$ im Flash-Speicher des Mobilgerätes. Durch diese Speichereinsparung sind $3\times$ mehr Grundrisse offline in einem Mobilgerät speicherbar. Auch die dynamische Belieferung mit entsprechenden Kacheln wird dadurch praktikabler: So kann über eine Verfallszeit die Cache-Lebensdauer von Karten gesteuert werden, womit aktuelle Informationen wie Baumaßnahmen oder gesperrte Räume in ein Echtzeit-Informationssystem aufgenommen werden können. Das dynamische Nachladen ermöglicht darüber hinaus Qualitätssteigerungen im Nachhinein.

Für eine zukünftige Version der Kartierungsapplikation ist eine Erweiterung auf die Unterstützung mehrerer Stockwerke geplant. Diese soll Nutzerinnen ermöglichen, in mehreren Ebenen durch ein Gebäude zu navigieren. Die Etagen sollen als weitere Overlays entweder automatisch abhängig von der Nutzerposition umschalten oder auch manuell umschaltbar sein, um eine Entdeckung eines gesamten Gebäudes „vom Sofa aus“ zu ermöglichen.

Gebäudegrundrisse sind für Nutzerinnen zoombar bis zum Erkennen einzelner Einrichtungsgegenstände. Insgesamt konnte die Aufösung im Vergleich zu den Herstellergrenzen um das 64-fache gesteigert werden.

Literatur

[AA16] Shurug Al-Khalifa und Muna Al-Razgan. „Ebsar: Indoor Guidance for the Visually Impaired“. In: *Computers & Electrical Engineering* 54 (Aug. 2016), S. 26–39. ISSN: 0045-7906. DOI: 10/gm4dd3

[Aka+18] Orhan Akal et al. „A Distributed Sensing Approach for Single Platform Image-Based Localization“. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. Dez. 2018, S. 643–649. DOI: 10/gm2tff

[App21] Apple. *iPhone 7 - Technische Daten*. https://support.apple.com/kb/SP743?locale=de_DE. (accessed 2022-06-17T12:53:23Z). Apr. 2021

[App22] Apple. *Extensions — The Swift Programming Language (Swift 5.7)*. https://docs.swift.org/swift-book/LanguageGuide/Extensions.html. (accessed 2022-06-17T09:23:28Z). Juni 2022

[App22] Apple. *MapKit*. https://developer.apple.com/tutorials/documentation/mapkit/. (accessed 2022-06-17T10:37:27Z). Apr. 2022

[App22] Apple. *The Longitude*. https://developer.apple.com/tutorials/documentation/corelocation/cllocationcoordinate2d/1423552-longitude/. (accessed 2022-06-07T13:55:01Z). Apr. 2022

[Aru+11] D. Arumugam et al. „Higher Order Loop Corrections for Short Range Magnetoquasistatic Position Tracking“. In: *2011 IEEE International Symposium on Antennas and Propagation (APSURSI)* (2011). DOI: 10/fm9phf

[BNH12] Joerg Blankenbach, Abdelmoumen Norrdine und Hendrik Hellmers. *A Robust and Precise 3D Indoor Positioning System for Harsh Environments*. Nov. 2012, S. 8. ISBN: 978-1-4673-1955-3. DOI: 10.1109/IPIN.2012.6418863

[Cer+20] Jesus D. Ceron et al. „Indoor Trajectory Reconstruction of Walking, Jogging, and Running Activities Based on a Foot-Mounted Inertial Pedestrian Dead-Reckoning System“. In: *Sensors* 20.3 (Jan. 2020), S. 651. DOI: 10/ggkwc5

[Cha+21] Neha Chaudhary et al. „An Indoor Visible Light Positioning System Using Tilted LEDs with High Accuracy“. In: *Sensors* 21.3 (Jan. 2021), S. 920. DOI: 10/gk28vf

[EPSG22] Klokan Technologies. *WGS 84 Pseudo-Mercator Projected Coordinate System for World*. http://epsg.io. (accessed 2022-06-10T13:19:08Z). Mai 2022

[Goo22] Google. *Location*. https://developer.android.com/reference/android/location/Location. (accessed 2022-06-07T13:42:47Z). Mai 2022

[Grz+20] Damian Grzechca et al. „How Accurate Can UWB and Dead Reckoning Positioning Systems Be? Comparison to SLAM Using the RPLidar System“. In: *Sensors* 20.13 (Jan. 2020), S. 3761. DOI: 10/gjzzmh

[Hes+16] Wolfgang Hess et al. „Real-Time Loop Closure in 2D LIDAR SLAM“. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Mai 2016, S. 1271–1278. DOI: 10/gfttjz

[Hua+15] Jiawei Huang et al. „IM6D: Magnetic Tracking System with 6-DOF Passive Markers for Dexterous 3D Interaction and Motion“. In: *ACM Transactions on Graphics* 34.6 (Okt. 2015), 217:1–217:10. ISSN: 0730-0301. DOI: 10/gjbt67

[Hua+21] Lu Huang et al. „A Real-Time Indoor Positioning System Based on Wi-Fi RTT and Multi-source Information“. In: *China Satellite Navigation Conference (CSNC 2021) Proceedings*. Hrsg. von Changfeng Yang und Jun Xie. Lecture Notes in Electrical Engineering. Singapore: Springer, 2021, S. 439–449. ISBN: 9789811631382. DOI: 10.1007/978-981-16-3138-2_41

[IEEE19] „IEEE Standard for Floating-Point Arithmetic“. In: *IEEE Std 754-2019 (Revision of IEEE 754-2008)* (Juli 2019), S. 1–84. DOI: 10.1109/IEEESTD.2019.8766229

[Kam22] Martin Kampf. *Geografische Koordinatensysteme*. https://www.kompf.de/gps/coords.html. (accessed 2022-06-08T09:29:35Z). März 2022

[KGA17] A. Khalajmehrabadi, Nikolaos Gatsis und D. Akopian. „Modern WLAN Fingerprinting Indoor Positioning Methods and Deployment Challenges“. In: *IEEE Communications Surveys & Tutorials* (2017). DOI: 10/gftg9s

[KRW15] Manfred Kässens, Rudolph Reimer und Rainer Wegerhoff. „Mikroskopische Verfahren“. In: *Romeis - Mikroskopische Technik*. Hrsg. von Maria Mulisch und Ulrich Welsch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, S. 1–42. ISBN: 978-3-642-55189-5. DOI: 10.1007/978-3-642-55190-1_1

[Kun17] Alexander Kunst. *Umfrage zum genutzten Smartphone-Betriebssystem in Deutschland 2017*. Techn. Ber. Statista, Apr. 2017

[LB21] Marius Laska und Jörg Blankenbach. „DeepLocBox: Reliable Fingerprinting-Based Indoor Area Localization“. In: *Sensors* 21.6 (März 2021), S. 2000. ISSN: 1424-8220. DOI: 10.3390/s21062000

[LDW18] Zhenyu Liu, Wenhan Dai und Moe Z. Win. „Mercury: An Infrastructure-Free System for Network Localization and Navigation“. In: *IEEE Transactions on Mobile Computing* 17.5 (Mai 2018), S. 1119–1133. ISSN: 1558-0660. DOI: 10.1109/TMC.2017.2725265

[Liu+20] Fei Liu et al. „The Indoor Localization of a Mobile Platform Based on Monocular Vision and Coding Images“. In: *ISPRS International Journal of Geo-Information* 9.2 (Feb. 2020), S. 122. DOI: 10/gm2snz

[LRS21] Bernhard Lahres, Gregor Raýman und Stefan Strich. *Objektorientierte Programmierung: das umfassende Handbuch*. 5., aktualisierte Auflage. Rheinwerk Computing. Bonn: Rheinwerk Verlag, 2021. ISBN: 978-3-8362-8317-5

[Lu+17] Guoyu Lu et al. „Indoor Localization via Multi-View Images and Videos“. In: *Computer Vision and Image Understanding* 161.C (Aug. 2017), S. 145–160. ISSN: 1077-3142. DOI: 10/gbx48v

[Mao+20] Pruttapon Maolanon et al. „Development of a Single Shot In-House SLAM Based Maps Construction Using Household Objects and Furnitures Information from CNNs Detector via Multi-Camera and Mobile Robot“. Thesis. Kasetsart University, Juli 2020

[Mic22] Microsoft. *GeoCoordinate.Longitude Property (System.Device.Location)*. https://docs.microsoft.com/en-us/dotnet/api/system.device.location.geocoordinate.longitude. (accessed 2022-06-07T13:52:05Z). Feb. 2022

[Mor+21] Dinis Moreira et al. „Human Activity Recognition for Indoor Localization Using Smartphone Inertial Sensors“. In: *Sensors* 21.18 (Sep. 2021), S. 6316. ISSN: 1424-8220. DOI: 10.3390/s21186316

[Mus21] Olaf Musch. *Design Patterns mit Java: Eine Einführung*. Wiesbaden: Springer Fachmedien Wiesbaden, 2021. ISBN: 978-3-658-35491-6. DOI: 10.1007/978-3-658-35492-3

[Paw90] James B. Pawley, Hrsg. *Handbook of Biological Confocal Microscopy*. Rev. ed. New York: Plenum Press, 1990. ISBN: 978-0-306-43538-6

[Pla+20] Benny Platte et al. „ARTranslate - Immersive Language Exploration with Object Recognition and Augmented Reality“. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, Mai 2020, S. 356–362

[PLP21] Soyoung Park, Jae Hong Lee und Chan Gook Park. „Robust Pedestrian Dead Reckoning for Multiple Poses in Smartphones“. In: *IEEE Access* 9 (2021), S. 54498–54508. ISSN: 2169-3536. DOI: 10/gjzzg9

[Rah+17] Thibaut Raharijaona et al. „Local Positioning System Using Flickering Infrared LEDs“. In: *Sensors* 17.11 (Nov. 2017), S. 2518. DOI: 10/gf8tb3

[Ram22] Frederik Ramm. *OpenStreetMap API v0.6 Documentation*. https://wiki.openstreetmap.org/wiki/API_v0.6. (accessed 2022-06-08T13:23:09Z). Apr. 2022

[Sha+18] Md. Shahjalal et al. „An Implementation Approach and Performance Analysis of Image Sensor Based Multilateral Indoor Localization and Navigation System“. In: *Wireless Communications and Mobile Computing* 2018 (Okt. 2018), S. 1–13. ISSN: 1530-8669, 1530-8677. DOI: 10/gfvtm2

[Tom22] TomTom. *Tomtom Maps SDK*. https://developer.tomtom.com/maps-ios-sdk/documentation/map-display/examples/map-events. (accessed 2022-06-07T14:45:14Z). Jan. 2022

[Tom22] Tomtom. *Zoom Levels and Tile Grid | Map Display API*. https://developer.tomtom.com/map-display-api/documentation/zoom-levels-and-tile-grid. (accessed 2022-05-14T09:37:14Z). März 2022

[Wik19] Wikipedia. „Separation of Concerns“. In: *Wikipedia* (Nov. 2019)

[Yan+15] Zheng Yang et al. „Mobility Increases Localizability: A Survey on Wireless Indoor Localization Using Inertial Sensors“. In: *ACM Computing Surveys* 47.3 (Apr. 2015), 54:1–54:34. ISSN: 0360-0300. DOI: 10/gf279x

[Zhe+20] Y. Zheng et al. „Magnetic-Based Positioning System for Moving Target With Feature Vector“. In: *IEEE Access* 8 (2020), S. 105472–105483. ISSN: 2169-3536. DOI: 10/gmxwhk