Praktikumsteil 3 - WiFi Location Fingerprinting

Dieses Projekt unterstützt Sie dabei, Implementierung und Auswertung eines WiFi-Positionierungssystems zu erlernen. Sie implementieren und experimentieren mit verschiedenen WiFi-Positionierungsmethoden zur Berechnung der Position eines WiFi-Gerätes in einem lokalen Koordinatensystem.

Ihre Implementierungsarbeit als auch Ihre Evaluationsergebnisse sollten Sie in einem kurzen Bericht darlegen (ca. 2 Seiten, exklusive Abbildungen). Der Bericht sollte eine kurze Beschreibung Ihrer Design- und Implementierungsentscheidungen enthalten und eine Beschreibung, wie die erstellten Programme verwendet werden können, z.B. wie Sie die Offline- und Online-Datasets angeben können.

Die erstellten Programme sollten Sie in lauffähiger Form auf dem SVN-Server des Kurses einstellen.

Der Bericht muss am Mittwoch 12.00 Uhr vor dem 3. Praktikumstermin per Mail abgegeben werden.

Hinweise: Das Praktikum wird nicht benotet, aber es kann Teilgegenstand während der mündlichen Prüfung am Ende des Semesters werden. Sie sollten dann in der Lage sein, Ihre Gruppenarbeit und die zugrunde liegende Theorien und Techniken bei der mündlichen Prüfung zu präsentieren.

Die Dateien MU.1.5meters.offline.trace und MU.1.5meters.online.trace enthalten WiFi-Messungen, gesammelt an der Universität Mannheim. Die Offline-Trace-Datei enthält WiFi-Messungen, die an Punkten gesammelt wurden, die 1,5 Meter voneinander entfernt in einem regulären Raster liegen. Das Raster ist durch blaue Punkte in MannheimDatasets.png visualisiert. Die Online-Trace-Datei enthält WiFi-Messungen, die an ausgewählten Punkten gesammelt wurden, die nicht mit einem der Gitterpunkte übereinstimmen, und die als purpurrote Punkte in MannheimDatasets.png visualisiert sind. Diese Traces sollten in Ihren Experimenten als Testdaten verwendet werden für die Positionen zu bestimmen sind. Die Koordinaten der APs sind in MU.AP.positions angegeben, und zwar bzgl. des Koordinatensystems, dass in MannheimDatasets.png visualisiert ist.

Das Archiv LocUtil.zip enthält Java-Quellcode, um Traces von WiFi-Messungen zu laden. Der Code in der Datei LocUtilExample.java gibt ein Beispiel dafür, wie der Code zum Laden einer Offline- und einer Online-Datei verwendet werden kann. Der Code erfordert viel Speicher und daher ist es notwendig, die Größe des Java-VM-Speichers mit zumindest -Xmx128m anzugeben. Der Tracegenerator ist randomisiert, so dass Sie nicht erwarten können, die gleichen Messungen in zwei verschiedenen Runs zu erhalten.

Teil 1

Sie sollten die folgenden Programme zur WiFi-Positionierung implementieren.

Obligatorischer Teil 1.a: Erstellen Sie ein Programm empirical_FP_NN, welches

empirisches Fingerprinting mit der matching method „nearest neighbor“ implementiert. Das Programm sollte eine Datei ausgeben, die für einen eingegebenen online-Trace in jeder Zeile eine geschätzte Position, zusammen mit der passenden tatsächlichen Position ausgibt.

Obligatorischer Teil 1.b: Analog zu 1.a erstellen Sie ein Programm model_FP_NN, das *modellbasiertes* Fingerprinting **(ohne WAF)** mit der matching method „nearest neighbor“ implementiert. Die Parameter des Modells sollten veränderbar sein.

Obligatorischer Teil 1.c: Analog zu obigem, erstellen Sie ein Programm empirical_FP_KNN, das empirisches Fingerprinting mit der matching method „*k-nearest neighbor*“ implementiert. Der Parameter k sollte veränderbar sein.

Obligatorischer Teil 1.d: Analog zu obigem, erstellen Sie ein Programm model _FP_KNN, das modellbasierte Fingerprinting **(ohne WAF)** mit der matching method „*k-nearest neighbor*“ implementiert. Der Parameter k sollte veränderbar sein.

Obligatorischer Teil 1.e: Erstellen Sie ein Programm scoreNN, das eine Ergebnisdatei liest und eine neue Datei mit einer kumulativen Verteilungsfunktion der Fehlerwerte erzeugt. Letzteres können Sie z.B. durch Sortieren der Fehlerwerte und anschließendes Errechnen, für jeden Fehlerwert, wie viele Prozent aller Fehlerwerte gleich oder kleiner als dieser Fehlerwert sind.

Allgemeine Überlegungen und Hinweise:
Für die modellbasierten Methoden können Parameterwerte aus den in Bahl et al. Für n und P (do). **Die folgend für den Mannheim-Datensatz angegebenen P (do) -Schätzungen gelten für Messungen d=1m entfernt vom AP: n: 3,415 , P (do): -33,77)**
Die folgenden Entscheidungen sollten Sie wohlüberlegt treffen, um optimale Ergebnisse zu erzielen:

- Sie müssen entscheiden, wie Sie in den nächste-Nachbar-Algorithmen handeln, dass Signalstärke-Messungen zu verschiedenen (wenn auch nahen) Orten oder Zeiten nicht immer die gleiche Menge am APs „hören“ können, d.h. deren Signalstärken messen können.
- Sie müssen entscheiden, wie sie unhörbaren APs modellieren, und, für die modellbasierte Methode, wann ein AP als unhörbar modelliert ist.
- Sie müssen entscheiden, wie Sie APs filtern, für die Sie keine Koordinaten im modellbasierten Algorithmus haben.
- Sie müssen entscheiden, wie der Abdeckungsbereich für den modellbasierten Algorithmus begrenzt werden soll.
- Sie müssen entscheiden, wie Messungen von mehreren Signalstärke-Proben für Fingerabdrücke und Online-Messungen zusammengefasst werden.
- Sie müssen entscheiden, über wieviele Online-Messungen Sie mitteln, um

eine einzelne Positionsbestimmung durchzuführen.

Teil 2

Im zweiten Teil dieses Projektes sollten Sie Experimente entwerfen und durchführen, um zu untersuchen, wie die Positionierungsgenauigkeit von Ihren Algorithmen und deren Parametern abhängt. Ihre Genauigkeis-Experimente sollten mindestens 10 Mal wegen der Zufälligkeit des Tracegenerators wiederholt und gemittelt werden. Sie sollen die folgenden Teil-aufgaben erfüllen:

Obligatorischer Teil 2.a: Ertellen Sie einen Plot, der in Bezug setzt die gemessenen Signalstärke an verschiedenen Messpunkten und die Entfernung vom jeweiligen Messpunkt zu den in der Messung „gehörten“ APs.

Obligatorischer Teil 2.b: Stellen Sie jeweils die kumulativen Distributionsfunktion (für alle erhaltenen Fehlerwerte) für jeden der vier obligatorischen Algorithmen 1.a-d; bei 1.c und 1.d wählen Sie hierfür k = 3.

Obligatorischer Teil 2.c: Vergleichen Sie für die beiden k-basierten Algorithmen 1.c und 1.d, die Median-Genauigkeit (Mittelwert über 10 Läufe) für verschiedene Werte von K (z. B. 1 bis 8).