Indoor Positioning Using the OpenHPS Framework

Maxim Van de Wynckel, Beat Signer

Web & Information Systems Engineering Lab
Vrije Universiteit Brussel
What is OpenHPS?

An Open Source Hybrid Positioning System

Data Frame

Data frames are envelopes that are transmitted and processed through a positioning model. These frames are created by source nodes (e.g., sensors) and contain one or more data objects needed to process the frame.

A frame should contain a single reading of a sensor (such as an image of a video stream or current acceleration) and not permanent or calculated information.

Creating data frames

OpenHPS is a framework that processes sensor information to retrieve a position for one or more data objects. These objects are contained within an envelope called a data frame.

```python
import { DataObject, DataFrame } from '@openhps/core';

const myObject = new DataObject('blender', 'Blender Signer');
const frame = new DataFrame();
frame.addObject(myObject);

(frame) DataFrame.addObject(object: DataObject): void
```

A basic data frame supports the addition of objects. Extended versions of this basic data frame also add additional sensor data.

Creating a custom data frame

Similar to data objects, decorators have to be used to indicate a serializable data frame.

```python
import {
    DataFrame,
    SerializableObject,
    SerializableReader
} from '@openhps/core';

@SerializableObject()
export class MyDataFrame extends DataFrame {
    public override @?image: any = undefined;
}
What is OpenHPS?

An Open Source Hybrid Positioning System

► Any technology
► Any algorithm
► Various use cases
► Flexible processing and output
  ■ Accuracy over battery consumption, reliability, ...
► Aimed towards
  ■ Developers
  ■ Researchers
Process Network Design

IMU Sensor
Process Network Design ...
Process Network Design ...

- IMU Source
- PDR Processing
- Merge Frames
- Display Position
- Data Service
- Load
- Save
Modularity

- Abstractions (e.g. location-based services)
  - Communication (e.g. socket connection)
  - Positioning Techniques (e.g. fingerprinting)
- Data Storage (e.g. MongoDB)
- Core Component
Modularity ...

Communication
Socket, MQTT, REST API, ...

Data Storage
MongoDB, LocalStorage, RDF, ...

Positioning Algorithms
IMU, fingerprinting, OpenVSLAM, ...

Abstractions
Geospatial, location-based services, geojson, ...

Other
React-Native, NativeScript, Sphero, ...
Data Processing

Knowledge

Raw Data

Processed Data
DataObject
Absolute and Relative Positions

**Absolute**
- 2D, 3D, Geographical, ...

**Relative**
- Distance, angle, velocity, ...
- Relative to another *object*
**Dataframe**

![Diagram of Dataframe]

- **VideoDataFrame**
  - `uid` and `timestamp` columns
  - `source`:
    - **CameraObject**
    - `uid`: "camera"
    - `position`: `{ x: 2, y: 5, z: 3 }
    - `projection`: ...
    - `width`: 1280
    - `height`: 1024

- **Image**

- **DataObject**
  - `Detected object`

- **DataObject**
  - `Detected object`

- **DataObject**
  - `Detected object`
DataFrame ...

Pushing Data

SourceNode  ProcessingNode  ProcessingNode  SinkNode

push(data)  push(data')  push(data'')

resolve push  resolve push  resolve push

completed data''  completed data''  completed data''
DataFrame ...

Pulling Data
Pushing Error

SourceNode: push(data) → resolve push → emit error

ProcessingNode: push(data') → resolve push → emit error

ProcessingNode: push(data'') → resolve push

SinkNode: 

reject push
SymbolicSpace

An object that semantically defines a space

- Spatial hierarchy
- Graph connectivity with other spaces
- Geocoding
- GeoJSON compatibility
- Can be used as a location
- Can be extended...
const building = new Building("PL9")
  .setBounds({
    topLeft: new GeographicalPosition(
      50.8203,
      4.3922),
    width: 46.275,
    height: 37.27,
    rotation: -34.04
  });

const floor = new Floor("PL9.3")
  .setBuilding(building)
  .setFloorNumber(3);

const office = new Room("PL9.3.58")
  .setFloor(floor)
  .setBounds([new Absolute2DPosition(4.75, 31.25),
              new Absolute2DPosition(8.35, 37.02),]);
Location-based Service

gGetCurrentPosition("me", ...)

Source → Processing

Source → Processing

Fusion

Sink

Location-based Service

Storage
Location-based Service ...

```java
setCurrentPosition("me", ...)
```
Location-based Service ...

watchPosition("me", ...)

Source → Processing

Fusion

Sink

Location-based Service

Storage
Demonstration

- Indoor positioning **use case**
- Use **existing techniques**
- Validation of **flexibility** and modularity
Positioning Model

- WiFi Source
- User Input
- BLE Source
- Merge Frames
- Socket Sink

Offline-stage App
Positioning Model...

Socket

Fingerprint

Storage

Sink

Socket Source

Source

Fingerprint

Service

Server

WLAN Fingerprint Processing

BLE Fingerprint Processing

Beacon Data Service

Fingerprint Service
Positioning Model ...
Positioning Model ...

- **Socket Source**
- **WLAN Fingerprint Processing**
- **Position Fusion**
- **Socket Sink**

**Beacon Data Service**

**Fingerprint Service**

**BLE Fingerprint Processing**

**BLE Multilateration Processing**

**Server**
Positioning Model ...
Positioning Model

Online App

PDR Processing

- **SMA Filter**: Simple moving average filter for acceleration
- **Gravity Processing**: Filter out gravity using orientation
- **Step Detection**: Adapt linear velocity based on detected step(s)
- **Velocity Processing**: Apply velocity to last known position
Positioning Model

Online App

```javascript
ModelBuilder.create()
  .addShape(GraphBuilder.create())
  .from(new IMUSourceNode(
      source: new DataObject(phoneUID),
      interval: 20,
      sensors: [
        SensorType.ACCELEROMETER,
        SensorType.ORIENTATION
      ]
    )))
  .via(new SMAFilterNode(
      frame => [frame, "acceleration"],
      { taps: 10 }
    ))
  .via(new GravityProcessingNode(
      method: GravityProcessingMethod.ABSOLUTE_ORIENTATION
    )))
```
Dataset ...

Total BLE Beacons: 11
Total detected WLAN access points: 220
Total stable WLAN access points: 199

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datapoints</td>
<td>110</td>
<td>30</td>
</tr>
<tr>
<td>Total fingerprints</td>
<td>440</td>
<td>120</td>
</tr>
<tr>
<td>Duration (per orientation)</td>
<td>20s</td>
<td>20s</td>
</tr>
<tr>
<td>Avg. WLAN Scans (per fingerprint)</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Avg. BLE Advertisements (per fingerprint)</td>
<td>16</td>
<td>15</td>
</tr>
</tbody>
</table>
## Validation Results

### Static Positioning

<table>
<thead>
<tr>
<th></th>
<th>WLAN fingerprinting</th>
<th>BLE fingerprinting</th>
<th>BLE multilateration</th>
<th>Fusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>failed points</td>
<td>0</td>
<td>6</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>average error</td>
<td>1.23 m</td>
<td>3.23 m</td>
<td>4.92 m</td>
<td>1.37 m</td>
</tr>
<tr>
<td>minimum error</td>
<td>0.01 m</td>
<td>0.17 m</td>
<td>0.74 m</td>
<td>0.01 m</td>
</tr>
<tr>
<td>maximum error</td>
<td>4.77 m</td>
<td>15.39 m</td>
<td>19.26 m</td>
<td>9.75 m</td>
</tr>
<tr>
<td>hit rate</td>
<td>95.82 %</td>
<td>80.83 %</td>
<td>52.50 %</td>
<td>96.67 %</td>
</tr>
</tbody>
</table>
Validation Results ...

Trajectories

- Sensor fusion
- WLAN & BLE Cell-ID
- Expected trajectory
- Trajectory start
Validation Results ...

Trajectories

<table>
<thead>
<tr>
<th></th>
<th>WLAN + BLE</th>
<th>WLAN + BLE + IMU</th>
</tr>
</thead>
<tbody>
<tr>
<td>average error</td>
<td>3.28 m</td>
<td>1.26 m</td>
</tr>
<tr>
<td>maximum error</td>
<td>9.60 m</td>
<td>3.10 m</td>
</tr>
<tr>
<td>average update frequency</td>
<td>3.04 s</td>
<td>0.52 s</td>
</tr>
</tbody>
</table>

![Diagram of trajectories with sensor fusion, WLAN & BLE Cell-ID, and expected trajectory]
Contributions and Conclusions

- OpenHPS: open source framework for hybrid positioning
  - Aimed towards developers and researchers
- Abstractions such as location-based services and spaces
- Validation of an indoor positioning use case
- Configurable and interchangeable nodes and services
- Public dataset with multiple orientations

Visit https://openhps.org for additional resources, documentation, source code and more!