OTB – Onderzoek voor
de gebouwde omgeving
Faculteit Bouwkunde

CLUSTERING AND INDEXING HISTORIC AIS DATA WITH SPACE FILLING CURVES

Martijn Meijers & Peter van Oosterom

January, 2018
GISt Report No. 72 – Version 1.1.1

TU Delft Delft University of Technology

Challenge the future

CLUSTERING AND INDEXING HISTORIC AIS DATA WITH SPACE FILLING CURVES

This project has been carried out in the framework of the “RWS-TUD Raamovereenkomst betreffende Samenwerking en Kennisuitwisseling op gebied van Ruimtelijke Informatievoorziening” (reference 31103836).

Authors:
Martijn Meijers & Peter van Oosterom
Contactperson Rijkswaterstaat: Wiebrand Bouwkamp

GISt Report No. 72 – Version 1.1.1

January, 2018

OTB – Research for the built environment
Faculty of Architecture and the Built Environment
Julianalaan 134, 2628 BL Delft
Tel. (015) 278 30 05
E-mail: OTB-bk@tudelft.nl
http://www.otb.bk.tudelft.nl
ISSN: 1569-0245
ISBN: 978-90-7702-941-1

Contents

Preface

Within Rijkswaterstaat (RWS) AIS (Automated Identification System) messages are received in real time with the DIAMONIS (Dutch Inland AIS monitoring Infrastructure) network. The current architecture of this system is not suited for archiving large amounts of historic AIS messages.

This report shows results of investigations into efficient storage for making these historic AIS messages available for a variety of use cases using the PostgreSQL Database extended with PostGIS. This is a follow-up research on research carried out in 2016. Space-filling curves are used for indexing and clustering the AIS position reports.

This project has been carried out in the framework of the "RWS-TUD Raamovereenkomst betreffende Samenwerking en Kennisuitwisseling op gebied van Ruimtelijke Informatievoorziening" (reference 31103836).

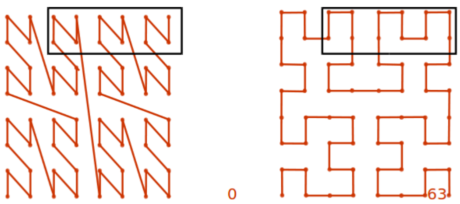
Chapter 1

Clustering and Indexing historic AIS data with Space Filling Curves

1.1 Introduction

- Earlier work gave an indication that Space Filling Curves (de Vreede, 2016) can be used as an indexing and clustering framework, creating 1 index where multiple columns are integrated (e.g. a space–time integrated index). The hypothesis is that this framework can be more effective for organizing geographic data with a time component than current state of the art techniques where each column of data will be indexed individually.
- However, this was not tested to its full potential with the database system used in that research (MongoDB).
- With another database system (e.g. PostgreSQL or Oracle) this is possible. However, these database systems do not have functions available to convert to and from Space-Filling curves.
- Hence, the objective of the research carried out is to make a SFC framework inside a Database Management System to verify the hypothesis.

1.2 Space filling curves



Peano (3 ranges) Hilbert (2 ranges)

- Studied Space Filling Curves (SFC), Morton and Hilbert curve:
  - Lawder (1999)
  - Lawder and King (2000)
  - Hamilton and Rau-Chaplin (2007)
  - Haverkort and Walderveen (2011)
  - Hamilton and Rau-Chaplin (2008)
  - Psomadaki (2016)

- Psomadaki et al. (2016)
- Bader (2013)

- Studied already existing implementations:
  - https://github.com/kwan2004/SFCLib
  - https://github.com/stpsomad/DynamicPCDMS
  - https://github.com/NLeSC/pointcloud-benchmark
  - http://www.tiac.net/~sw/2008/10/Hilbert
  - http://pdebuyl.be/blog/2015/hilbert-curve.html
- Current solutions were not available as extension to the database.
- As proof of principle, we therefore developed:
  - Functions for encoding and decoding in Python and Rust for both Morton and Hilbert in n dimensions (2D, 3D, 4D, ..., nD).
  - Functions for overlap testing of a nD geometry by means of n-ary tree traversal in Python and Rust (Psomadaki, 2016).
  - Made available a glue layer for calling the Python/Rust code from within the database using SQL.

1.3 AIS data loading

- Historic AIS data was obtained, structured as NMEA sentences stored hourly in a text file. Data for 1 year (September 2016 – August 2017) was used during the research (in total 300 GB of raw NMEA messages).
- A small program was created for loading the data into PostgreSQL. The program was created using the Rust programming language (as this is a compiled language, it offers fast execution).
- The data were filtered to obtain only the position reports (AIS message type: 1, 2, 3). The timestamp (in UTC time) and payload for every message was loaded into the database.
- From the source data 3 months of data was selected for loading: 1,069,908,800 AIS position reports were loaded.

- The payload was loaded into a bitvector data type as suggested by our previous research (Meijers et al., 2016, 2017). Also functional indexes were created on the table with the loaded data.
- Some database functions were made for obtaining the parameters latitude and longitude from the payload string.
- This table serves as heap table from which it is easy to make selections and load data in alternative forms (i.e. with and without using space filling curves).
- With QGIS the heap table can be visualised and queried.

1.4 Tests

We loaded data for comparison:

- From heap table made a table with: position (ϕ, λ), timestamp and original payload into a table with 2 separate indexes (one on space, one on time). Clustered the table on timestamp or on space index.
- From heap table made a table with: Space Filling Curve key and original payload into a table and add 1 integrated Space Filling Curve index (using Hilbert SFC). Cluster the table on SFC index.

Measure data storage size. Query with some different sized shaped boxes. See if query time gives different results and is in favour of one model or the other. Section 4.2 gives details.

1.5 Future work

- Keep metadata for how curve is positioned (scaling and translation of original data) inside a table inside the database, as this information is needed for posing queries to the data.
- Web Feature Service (WFS) service interface in front of the database, so to make it possible to connect to the database with the SFC model inside the database and a client application not knowing anything of the model, still able to fetch parts of the data (e.g. QGIS showing a certain geographic region with vessels at a certain time).
- Use instead of bigint (limited to storing integers with maximum 2^{63}, with e.g. 4 dimensions this limits the maximum range to $2^{63/4} = 32{,}768$) a data type to be able to produce full resolution SFC keys (e.g. varchar, bit vector).
- Implement more query geometries than only a nD-box.
- Compare different systems where SFC framework can also be implemented (e.g. Oracle with Index Organized Table and big numbers available, Cassandra).

Chapter 2

AIS message loading

2.1 Transform NMEA messages to Bitvector for loading

A small Rust[1] program for loading AIS messages into the PostgreSQL database by means of `COPY FROM`. PostgreSQL and PostGIS need to be installed. We used PostGIS 2.4.2 installed on PostgreSQL 10.1.

```
extern crate lzma;
extern crate glob;

use std::io::prelude::*;
use std::fs::File;
use std::io::{self, Read, Write};
use std::env;
use std::process;
use glob::glob;
use lzma::{LzmaReader, LzmaError};
use std::collections::HashMap;

fn main() {

    // the bit pattern that every character makes
    let mut bit6 = HashMap::new();
    bit6.insert('0', "000000");
    bit6.insert('1', "000001");
    // ... snipped ...
    bit6.insert('v', "111110");
    bit6.insert('w', "111111");

    let table_nm = "rws_ais_bits";
    println!("DROP TABLE IF EXISTS {};", table_nm);
    println!("CREATE TABLE {}(ts timestamp with time zone, payload bit varying)
        WITH OIDS;", table_nm);
    println!("COPY {}(ts, payload) FROM STDIN DELIMITER '\t';", table_nm);

    let mut ct = 0;
    let pattern = "*.xz";
    let dir = "/var/tmp/ais/nmea_2016/";
    let together = &format!("{}{}", dir, pattern);
    for entry in glob(together).unwrap() {
        let mut ict = 0;
        match entry {
            Ok(path) => {
                let f = File::open(path).unwrap();
                let mut reader = LzmaReader::new_decompressor(f).unwrap();
                let mut s = String::new();
                reader.read_to_string(&mut s).unwrap();
                for line in s.lines() {
                    let splitted = line.split(",");
                    let tmp: Vec<&str> = splitted.collect();
```

[1] https://www.rust-lang.org/

```
                    let first_char = tmp[5].chars().next();
                    if (first_char == Some('1') || first_char == Some('2') ||
                        first_char == Some('3')) && tmp[5].len() == 28 {
                            let mut outcome:Vec<&str> = Vec::new();
                            for ch in tmp[5].chars() {
                                let value = bit6.get(&ch);
                                match value {
                                    Some(x) => outcome.push(x),
                                    _ => {}
                                }
                            }
                            println!("{}Z\t{}", tmp[7], outcome.join(""));
                    }
                    ict += 1;
                }
            },
            // if the path matched but was unreadable,
            // thereby preventing its contents from matching
            Err(e) => panic!("{:?}", e),
        }
        ct += ict;
    }
    println!("\\.");
}
```

2.2 Functions for obtaining message fields

```
-- Message type as integer
--------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION ais_type(payload bit varying) RETURNS integer AS $$
        BEGIN
                RETURN substring(payload from 1 for 6)::integer;
        END;
$$ LANGUAGE plpgsql
IMMUTABLE
;

-- MMSI number as integer
--------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION ais_mmsi(payload bit varying) RETURNS integer AS $$
        BEGIN
                RETURN substring(payload from 9 for 30)::integer;
        END;
$$ LANGUAGE plpgsql
IMMUTABLE
;

-- Functions that deal with geometry
--------------------------------------------------------------------------------
-- Note, 1 more bit for Easting (as -90 +90 is smaller than -180 +180)

DROP FUNCTION ais_easting(bit varying) CASCADE;
DROP FUNCTION ais_northing(bit varying) CASCADE;
DROP FUNCTION ais_point(payload bit varying) CASCADE;

-- Northing / Latitude / Y
CREATE OR REPLACE FUNCTION ais_northing(payload bit varying) RETURNS real AS $$
        DECLARE
            result integer;
        BEGIN
            result := substring(payload from 90 for 27)::integer;
            IF (result & 67108864) > 0 THEN         -- pow(2, 27-1)::integer
                -- negative
                result := -(134217728 - result);    -- 2**27
            END IF;
            RETURN round(result / 600000.0, 5);
        END;
$$ LANGUAGE plpgsql
```

```
IMMUTABLE
;

-- Easting / Longitude / X
CREATE OR REPLACE FUNCTION ais_easting(payload bit varying) RETURNS real AS $$
        DECLARE
            result integer;
        BEGIN
            result := substring(payload from 62 for 28)::integer;
            IF (result & 134217728) > 0 THEN        -- pow(2, 28-1)::integer
                -- negative
                result := -(268435456 - result);    -- 2**28
            END IF;
            RETURN round(result / 600000.0, 5);
        END;

$$ LANGUAGE plpgsql
IMMUTABLE
;

-- Both N+E, combined in PostGIS point type (for OGC Simple Feature Access)
CREATE OR REPLACE FUNCTION ais_point(payload bit varying)
RETURNS geometry(Point, 4326) AS $$
        BEGIN
                RETURN st_setsrid(st_makepoint(ais_easting(payload),
                                               ais_northing(payload)),
                                  4326)::geometry(Point, 4326);
        END;
$$ LANGUAGE plpgsql
IMMUTABLE
;
```

2.3 Making functional indexes on the messages

```
\timing on

-- functional index on mmsi
--------------------------------------------------------------------------------
CREATE INDEX
    i__rws_ais_bits__ais_mmsi
ON
    rws_ais_bits (ais_mmsi(payload))
TABLESPACE
    indx
;

-- functional index on message type
--------------------------------------------------------------------------------
CREATE INDEX
    i__rws_ais_bits__ais_type
ON
    rws_ais_bits(ais_type(payload))
TABLESPACE
    indx
;

-- index on timestamps of messages
--------------------------------------------------------------------------------
CREATE INDEX
    i__rws_ais_bits__ts
ON
    rws_ais_bits(ts asc)
TABLESPACE
    indx
;

-- functional index on geometry
--------------------------------------------------------------------------------
CREATE INDEX
    i__rws_ais_bits__geometry
```

```
ON
    rws_ais_bits
USING GIST
    (ais_point(payload))
TABLESPACE
    indx
;

-- view
--------------------------------------------------------------------------------
create view v_rws_ais_bits_geom
as
select
    oid,
    ts,
    ais_mmsi(payload) as mmsi,
    ais_point(payload) as geometry
from
    rws_ais_bits
;

-- what's the size?
--------------------------------------------------------------------------------
SELECT
        pg_size_pretty (
              pg_relation_size ('rws_ais_bits'))
;
```

Chapter 3

Space Filling Curves inside DBMS

3.1 Implementation

The implementation is available at:

- https://bitbucket.org/bmmeijers/pysfc (Python library)
- https://bitbucket.org/bmmeijers/sfc-rs (Rust library)
- https://bitbucket.org/bmmeijers/sfc-rs-ffi (Python binding to Rust library)

3.2 Making Python functions available

Functions using PL/PYTHON for making Space Filling Curves inside the PostgreSQL DBMS available.

```
-- Hilbert
--------------------------------------------------------------------------------
-- Encode a nD Coordinate as Hilbert value
CREATE OR REPLACE FUNCTION hencode (arr integer[])
  RETURNS bigint
AS $$
    import sfc.hilbert
    return sfc.hilbert.encode(arr)
$$ LANGUAGE plpython2u;

-- Decode a Hilbert value as nD Coordinate
CREATE OR REPLACE FUNCTION hdecode (val bigint, dims integer)
  RETURNS integer[]
AS $$
    import sfc.hilbert
    return sfc.hilbert.decode(val, dims)
$$ LANGUAGE plpython2u;

-- N-order
--------------------------------------------------------------------------------
-- Encode a nD Coordinate as N-order value
CREATE OR REPLACE FUNCTION nencode (arr integer[])
  RETURNS bigint
AS $$
    import sfc.morton_norder
    return sfc.morton_norder.encode(arr)
$$ LANGUAGE plpython2u;

-- Decode a N-order value as nD Coordinate
CREATE OR REPLACE FUNCTION ndecode (val bigint, dims integer)
  RETURNS integer[]
AS $$
    import sfc.morton_norder
    return sfc.morton_norder.decode(val, dims)
```

```
$$ LANGUAGE plpython2u;

-- Querying
--------------------------------------------------------------------------------
-- Create ranges for Hilbert range search (join query)
DROP FUNCTION sfc_hquery;
CREATE OR REPLACE FUNCTION
    sfc_hquery(lo integer[], hi integer[])
        RETURNS
    TABLE(lower bigint, upper bigint)
AS $$
    import sfc.relate
    import sfc.query_hilbert
    return sfc.query_hilbert.hquery(query=sfc.relate.ndbox(lo, hi))
$$ LANGUAGE plpythonu;

-- Create ranges for N-order range search (join query)
DROP FUNCTION sfc_nquery;
CREATE OR REPLACE FUNCTION
    sfc_nquery(lo integer[], hi integer[])
        RETURNS
    TABLE(lower bigint, upper bigint)
AS $$
    import sfc.relate
    import sfc.query_norder
    return sfc.query_norder.nquery(query=sfc.relate.ndbox(lo, hi))
$$ LANGUAGE plpythonu;
```

3.3 Creating a table with SFC values

- Table creation

```
create table rws_3months_sfc as select * from (

with transform_params as
(
select
        sfc_transform_scale(-180, 180, 0, pow(2,21)::numeric) as scale_east,
        sfc_transform_scale( -90,  90, 0, pow(2,21)::numeric) as scale_north,
        sfc_transform_scale(
            extract(epoch from '2016-09-30 23:59:59+02'::timestamp with time zone)
                ::int,
            extract(epoch from '2016-12-31 23:59:59+02'::timestamp with time zone)
                ::int,
            0,
            pow(2,21)::numeric) as scale_time,
        -180::numeric as translate_east,
        -90::numeric as translate_north,
        extract(epoch from '2016-09-30 23:59:59+02'::timestamp with time zone)::
            numeric as translate_time
)

select
hencode(
  array[
    sfc_transform_dim(e, (select translate_east  from transform_params), (select
        scale_east  from transform_params))::int,
    sfc_transform_dim(n, (select translate_north  from transform_params), (select
        scale_north  from transform_params))::int,
    sfc_transform_dim(t, (select translate_time  from transform_params), (select
        scale_time  from transform_params))::int
  ]
) as hilbert,
ts,
pt,
payload
from (
    select
```

```
    extract(epoch from ts)::numeric as t,
    ais_easting(payload)::numeric as e,
    ais_northing(payload)::numeric as n,
    ts,
    ais_point(payload) as pt,
    payload
    from rws_ais_bits where
    ts between '2016-10-01 00:00:00+02'::timestamp with time zone and '2016-12-31
        23:59:00+02'::timestamp with time zone
    and
    ais_easting(payload) between -180 and 180
    and
    ais_northing(payload) between -90 and 90
) staging

) data
;
```

- Indexing

```
create index i__rws_3months_sfc__hilbert on rws_3months_sfc (hilbert) tablespace
    indx;
```

- Cluster + analyze on the table

```
cluster rws_3months_sfc using i__rws_3months_sfc__hilbert;
vacuum analyze rws_3months_sfc;
```

3.4 Querying

- Making a query

```
select * from (
with transform_params as (
    select
        sfc_transform_scale(-180, 180, 0, pow(2,21)::numeric) as scale_east,
        sfc_transform_scale( -90,  90, 0, pow(2,21)::numeric) as scale_north,
        sfc_transform_scale(1475272799, 1483225200, 0, pow(2,21)::numeric) as
            scale_time,
        -180 as translate_east,
        -90 as translate_north,
        1475272799.0 as translate_time)
select d.* from sfc_hquery(
    array[
            floor(sfc_transform_dim(3, (select translate_east  from
                transform_params), (select scale_east  from transform_params)))::
                int,
            floor(sfc_transform_dim(51, (select translate_north from
                transform_params), (select scale_north from transform_params)))::
                int,
            floor(sfc_transform_dim(extract(epoch from '2016-09-30 23:59:59+02'::
                timestamp with time zone)::int, (select translate_time  from
                transform_params), (select scale_time  from transform_params)))::
                int
    ],
    array[
            ceil(sfc_transform_dim(4, (select translate_east  from
                transform_params), (select scale_east  from transform_params)))::
                int,
            ceil(sfc_transform_dim(52, (select translate_north from
                transform_params), (select scale_north from transform_params)))::
                int,
            ceil(sfc_transform_dim(extract(epoch from '2016-10-01 23:59:59+02'::
                timestamp with time zone)::int, (select translate_time  from
                transform_params), (select scale_time  from transform_params)))::
                int
```

```
    ],
    14 -- how deep to descend
) as r,
rws_1day d
where d.hilbert >= r.lower and d.hilbert < r.upper
)
as result;
```

Chapter 4

Presentations

4.1 Plan

Presentation giving an overview of the planned research.

Research plan

Clustering and indexing AIS data with Space Filling Curves

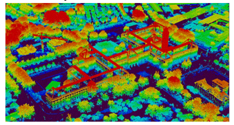
Martijn Meijers   Peter van Oosterom

Monday, October 30, 2017

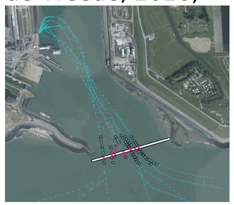
TUDelft

Introduction

- GIS technology
- Research topics – Geo-DBMS (Database Management System)
  - Point clouds: AHN2, sand motor (Psomadaki et al., 2016)

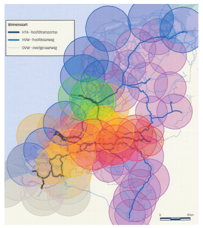


  - Moving objects – vessels / ... (Meijers et al., 2016; de Vreede, 2016)



Rijkswaterstaat

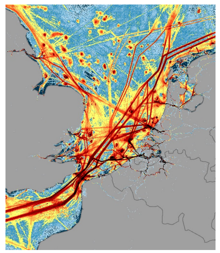
- AIS data: DIAMONIS* network (inland AIS)
- Large data volume: 1.5 GB data per week (80,000,000 messages)
- Historical data versus (near) real time + history
- Focus: Historical data only (+ bulkload)
- (Privacy aspects)



*Dutch Inland AIS monitoring Infrastructure

Use cases for historical data

- Analyse use/intensity of the fair way
- Locks / Passings: is it really crowded?
- Accidents: who was there and moving/not-moving?
- Distances between vessels: dangerous situations?
- AIS Network: where can reception be improved?
- ...

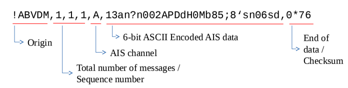


2 basic questions for many use cases

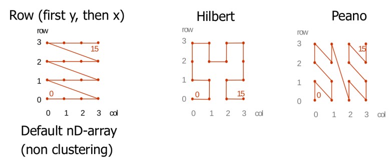
- Which vessels are where (inside rectangle) at given time? (location query)
- What trajectory did a vessel travel over given period? (trajectory query)
- 4D Pointcloud: Position (ϕ, λ), Identity, Timestamp

Storage

- Archive: Store all AIS messages
- This project: Only position messages (1, 2, 3)

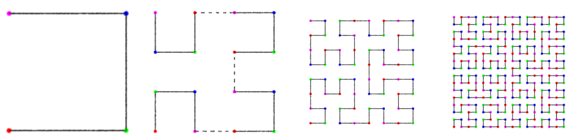


- Perform queries within reasonable time
- Need for: indexing and clustering
- Can Space Filling Curves help?



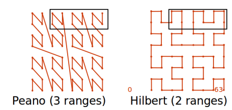
Space Filling Curves (SFC)

- A SFC runs through a nD space and with given resolution 'touch' all points of this space
- Unravel nD space into 1D: B-tree indexing
- Different types: Peano/Morton, Hilbert, . . .
- Subsequent points on curve are close in space: Clustering



Research plan

- Calculate SFC key (position on curve) and load AIS messages + key in Geo-DBMS
  - Oracle (Index Organized Table)
  - PostgreSQL + PostGIS (B-tree on key + Cluster)
- Perform relevant queries (Location + Trajectory) – Translate query geometry to SFC ranges



- Benchmark
  - Storage space
  - Query execution time for SFC model
  - Compare to separate indexes (Meijers et al., 2016)

Challenges ahead

- Which items to put in SFC key?
  - Position (ϕ, λ)
  - Identity
  - Timestamp
- How to scale the items their ranges? (data needs to be 'cube'-like for SFC, different scaling → different clustering)

Questions?

- dr.ir. Martijn Meijers
  b.m.meijers@tudelft.nl
  `http://www.gdmc.nl/martijn/`
  tel. (+31) 15 27 856 42
- Delft University of Technology
  Faculty of Architecture and the Built Environment
  OTB – Research for the built environment
  GIS Technology

References

de Vreede, I. (2016). Managing historic Automatic Identification System data by using a proper database management system structure. Master's thesis, Delft University of Technology.

Meijers, M., van Oosterom, P., and Quak, W. (2016). Management of AIS messages in a Geo-DBMS. Technical report, Delft University of Technology.

Psomadaki, S., van Oosterom, P., Tijssen, T., and Baart, F. (2016). Using a space filling curve approach for the management of dynamic point clouds. In Dimopoulou, E. and van Oosterom, P., editors, *ISPRS Annals Volume IV-2/W1, 11th 3D Geoinfo Conference*, pages 107–118. Athens.

4.2 Results

Presentation giving an overview of the research result.

Research results

Clustering and indexing AIS data with Space Filling Curves

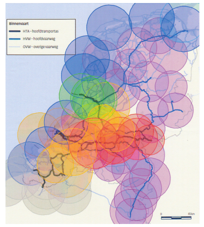
Martijn Meijers   Peter van Oosterom

Monday, January 22, 2018

TUDelft

Rijkswaterstaat

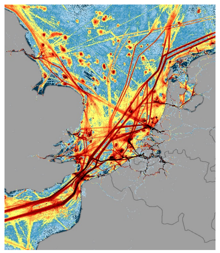
- AIS data: DIAMONIS* network (inland AIS)
- Large data volume: 1.5 GB data per week (80,000,000 messages)
- Historical data versus (near) real time + history
- Focus: Historical data only (+ bulkload)
- (Privacy aspects)



*Dutch Inland AIS monitoring Infrastructure

Use cases for historical data

- Analyse use/intensity of the fair way
- Locks / Passings: is it really crowded?
- Accidents: who was there and moving/not-moving?
- Distances between vessels: dangerous situations?
- AIS Network: where can reception be improved?
- ...

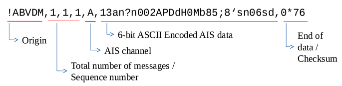


2 basic questions for many use cases

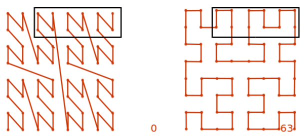
- **Which vessels are where (inside rectangle) at given time? (location query)**
- What trajectory did a vessel travel over given period? (trajectory query)
- 4D Pointcloud: Position (ϕ, λ), Identity, Timestamp

Storage

- Archive: Store all AIS messages
- This project: Only position messages (1, 2, 3)



- Perform queries within reasonable time
- Need for: indexing and clustering
- Can Space Filling Curves help?



Peano (3 ranges) Hilbert (2 ranges)

Loading data

- Test data as NMEA messages (text) + timestamp
- Load data as bit varying (bit vector) type in PostgreSQL
- Selected: Message types 1, 2, 3
- Implemented in Rust
- Load time indication: 3h52m for 4.3×10^9 messages
- Data for October 2016 – October 2017
- Subset for tests
  - 1,069,908,800 messages: Oct – Dec 2016
  - 12,106,659 messages: Oct 1, 2016

Functions for decoding AIS messages inside the DBMS

- Decoding messages implemented as PL/PgSQL functions inside the database

Functions for decoding AIS messages inside the DBMS

```
CREATE OR REPLACE FUNCTION
  ais_type(payload bit varying)
RETURNS integer AS $$
  BEGIN
    RETURN
      substring(payload from 1 for 6)::integer;
  END;
$$ LANGUAGE plpgsql
IMMUTABLE;
```

Functions for decoding AIS messages inside the DBMS

```
SELECT
  ais_type(payload),
  payload
FROM
  rws_ais_bits_small
LIMIT 1;
```


Functions for decoding AIS messages inside the DBMS

```
SELECT
  ais_type(payload),
  payload
FROM
  rws_ais_bits_small
LIMIT 1;

 ais_type  | payload
-----------+-----------------------------
        1  | 0000010000111...00111111110
```


Functions for decoding AIS messages inside the DBMS

```
CREATE OR REPLACE FUNCTION
  ais_easting(payload bit varying)
RETURNS real AS $$ DECLARE
  result integer;
BEGIN
  result := substring(payload
                      from 62 for 28)::integer;
  IF (result & 134217728) > 0 THEN   -- 2**(28-1)
    result := -(268435456 - result); -- 2**28
  END IF;
  RETURN round(result / 600000.0, 5);
END;
$$ LANGUAGE plpgsql
IMMUTABLE;
```


Functions for decoding AIS messages inside the DBMS

```
SELECT
  ais_easting(payload)
FROM
  rws_ais_bits_small
LIMIT 1;


 ais_easting
-------------
   4.1943302
```

Functions for decoding AIS messages inside the DBMS

```
geodata=# \d rws_ais_bits
          Table "rws_ais_bits"
 Column  |           Type           | Modifiers
---------+--------------------------+-----------
 ts      | timestamp with time zone | not null
 payload | bit varying              |
```

- Loaded table as heap table
- Select from this table data + add columns (e.g. place on Space Filling Curve)

SFC keys

- Conversion to Hilbert / Morton (N-order) key and back
- Implemented (as Python and Rust library + Python bindings[†])

```
hencode( (x, y, ...) ) -> key
hdecode( key, nD ) -> (x, y, ...)

nencode( (x, y, ...) ) -> key
ndecode( key, nD ) -> (x, y, ...)
```

- PL/Python functions in database
- Makes it possible to use inside SQL

[†] https://bitbucket.org/bmmeijers/pysfc
https://bitbucket.org/bmmeijers/sfc-rs
https://bitbucket.org/bmmeijers/sfc-rs-ffi

SFC keys

Timings for conversion from nD-coordinate to key and back

Time needed for encoding + decoding

	Encode + decode (s)	Ops / Sec
Python N-order	11.9	0.19M
Python Hilbert	30.7	0.07M
Rust+Python N-order	1.5	1.5M
Rust+Python Hilbert	1.9	1.2M

For 1.1M (x,y, ...) → key (and back)

Experiment

- ‘Heap table’ and database functions allow constructing test data sets
- Store 1 day of data in 2 tables (with and without SFC; SFC based on 2D space and 1D time)
- Index the tables
- Compare Table sizes
- Compare Query execution times

Table with separate indexes

```
geodata=# \d rws_1day_sep
          Table "rws_1day_sep"
 Column  |           Type           | Modifiers
---------+--------------------------+-----------
 ts      | timestamp with time zone |
 pt      | geometry                 |
 payload | bit varying              |
Indexes:
    "i__rws_1day_sep__pt" gist (pt) CLUSTER
    "i__rws_1day_sep__ts" btree (ts)
```


Table with SFC index

```
geodata=# \d rws_1day_sfc
            Table "rws_1day_sfc"
 Column  |           Type           | Modifiers
---------+--------------------------+-----------
 hilbert | bigint                   |
 ts      | timestamp with time zone |
 pt      | geometry                 |
 payload | bit varying              |
Indexes:
    "i__rws_1day_sfc__hilbert"
        btree (hilbert) CLUSTER
```


Table size comparison

Table + Indexes: Size (in MB)

	Separate	SFC	Δ SFC - Separate
Table	1075	1168	+93 ‡
Index	259 (ts) 614 (pt)	259 (key)	-614
Σ	1948	1427	-521

‡SFC key

Query Execution (separate indexes)

Query steps:

- Determine nD-region (timespan + space)
- Formulate SQL query
- Query planner decides: which index is most selective (space **or** time)
- Database retrieves data using selected index and prunes results which additional filter step on other dimension

```
select d.* from rws_1day_sep d
where
  ts between '2016-09-30 23:59:59+02'
      and '2016-10-01 23:59:59+02'
and
  st_makebox2d(3, 51, 4, 52) && pt;
```

Query Execution (integrated, SFC)

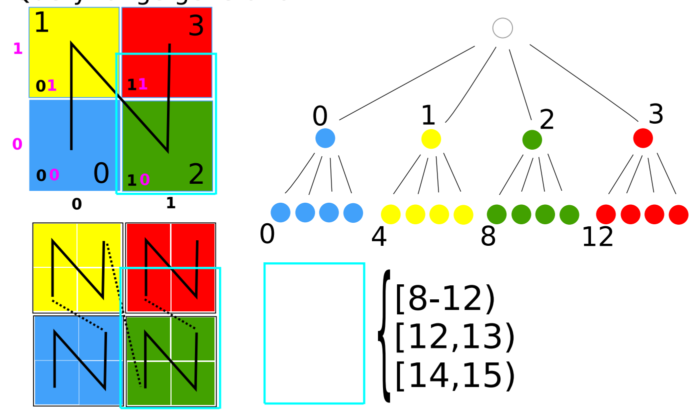
Query steps:

- Determine nD-region (timespan + space)
- Translate query geometry in SFC ranges (using SQL functions, that positions 'cube') — Quadtree, Octree, n-ary tree
- Join range table to data table (use index on SFC key!)
- Filter result on exact geometry

```
select d.* from sfc_hquery(
    array[east_min, north_min, time_min],
    array[east_max, north_max, time_max],
    14,  -- maximum level in n-ary tree to descend
) as r
join
    rws_1day_sfc d
on
    d.hilbert >= r.lower and d.hilbert < r.upper;
```


Query Execution (integrated, SFC)

Query range generation



Query Execution comparison

- Separate model: Database has to choose index
- Integrated model: Once ranges are produced, index on data table is used
- Challenge for integrated model: tune selectivity (number of ranges for query – how deep to descend tree)
- Limited experiments, but query performance integrated model on-par with separate model (<1-2 secs).

Conclusions

- Integrated SFC approach works
- Needs less storage
- Needs tuning for performing queries, but on-par with separate index queries

Future work

- More query performance tests (different parameters: tree depth, scaling of domain, compare number of records retrieved for exact versus approximation, . . .)
- Better compare Morton versus Hilbert: Hilbert more expensive to compute, but less separate ranges for queries (observed): Noticable?
- Instead of Axis-Aligned Box Query geometry also other shapes: e.g. Circle / Line. . . (integrated SFC approach is expected to perform very good initial filtering if ‘lots of empty space’ in the query geometry)
- Integrate more dimensions: Needs large number types/storage
- Compare with other systems, e.g. Oracle (IOT, large numbers)

Questions?

- dr.ir. Martijn Meijers
  b.m.meijers@tudelft.nl
  `http://www.gdmc.nl/martijn/`
  tel. (+31) 15 27 856 42
- Delft University of Technology
  Faculty of Architecture and the Built Environment
  OTB – Research for the built environment
  GIS Technology

Bibliography

Bader, M. (2013). *Space-Filling Curves*. Springer Berlin Heidelberg.

de Vreede, I. (2016). Managing historic Automatic Identification System data by using a proper database management system structure. Master's thesis, Delft University of Technology.

Hamilton, C. H. and Rau-Chaplin, A. (2007). Compact hilbert indices for multi-dimensional data. In *First International Conference on Complex, Intelligent and Software Intensive Systems (CISIS'07)*. IEEE.

Hamilton, C. H. and Rau-Chaplin, A. (2008). Compact hilbert indices: Space-filling curves for domains with unequal side lengths. *Information Processing Letters*, 105(5):155–163.

Haverkort, H. and Walderveen, F. V. (2011). Four-dimensional hilbert curves for r-trees. *Journal of Experimental Algorithmics*, 16:3.1.

Lawder, J. (1999). *The Application of Space-filling Curves to the Storage and Retrieval of Multi-dimensional Data*. PhD thesis, University of London.

Lawder, J. K. and King, P. J. H. (2000). Using space-filling curves for multi-dimensional indexing. In *Lecture Notes in Computer Science*, pages 20–35. Springer Berlin Heidelberg.

Meijers, M., Quak, W., and van Oosterom, P. (2017). Archiving AIS messages in a Geo-DBMS. In Bregt, A., Sarjakoski, T., van Lammeren, R., and Rip, F., editors, *Proceedings of the 20th AGILE Conference on Geographic Information Science*, pages 1–3.

Meijers, M., van Oosterom, P., and Quak, W. (2016). Management of AIS messages in a Geo-DBMS. Technical report, Delft University of Technology.

Psomadaki, S. (2016). Using a Space Filling Curve for the Management of Dynamic Point Cloud Data in a Relational DBMS. Master's thesis, Delft University of Technology.

Psomadaki, S., van Oosterom, P., Tijssen, T., and Baart, F. (2016). Using a Space Filling Curve Approach for the Management of Dynamic Point Clouds. In Dimopoulou, E. and van Oosterom, P., editors, *ISPRS Annals Volume IV-2/W1, 11th 3D Geoinfo Conference*, pages 107–118. Athens.

Reports published before in this series

1. GISt Report No. 1, Oosterom, P.J. van, Research issues in integrated querying of geometric and thematic cadastral information (1), Delft University of Technology, Rapport aan Concernstaf Kadaster, Delft 2000, 29 p.p.
2. GISt Report No. 2, Stoter, J.E., Considerations for a 3D Cadastre, Delft University of Technology, Rapport aan Concernstaf Kadaster, Delft 2000, 30.p.
3. GISt Report No. 3, Fendel, E.M. en A.B. Smits (eds.), Java GIS Seminar, Opening GDMC, Delft 15 November 2000, Delft University of Technology, GISt. No. 3, 25 p.p.
4. GISt Report No. 4, Oosterom, P.J.M. van, Research issues in integrated querying of geometric and thematic cadastral information (2), Delft University of Technology, Rapport aan Concernstaf Kadaster, Delft 2000, 29 p.p.
5. GISt Report No. 5, Oosterom, P.J.M. van, C.W. Quak, J.E. Stoter, T.P.M. Tijssen en M.E. de Vries, Objectgerichtheid TOP10vector: Achtergrond en commentaar op de gebruikersspecificaties en het conceptuele gegevensmodel, Rapport aan Topografische Dienst Nederland, E.M. Fendel (eds.), Delft University of Technology, Delft 2000, 18 p.p.
6. GISt Report No. 6, Quak, C.W., An implementation of a classification algorithm for houses, Rapport aan Concernstaf Kadaster, Delft 2001, 13.p.
7. GISt Report No. 7, Tijssen, T.P.M., C.W. Quak and P.J.M. van Oosterom, Spatial DBMS testing with data from the Cadastre and TNO NITG, Delft 2001, 119 p.
8. GISt Report No. 8, Vries, M.E. de en E. Verbree, Internet GIS met ArcIMS, Delft 2001, 38 p.
9. GISt Report No. 9, Vries, M.E. de, T.P.M. Tijssen, J.E. Stoter, C.W. Quak and P.J.M. van Oosterom, The GML prototype of the new TOP10vector object model, Report for the Topographic Service, Delft 2001, 132 p.
10. GISt Report No. 10, Stoter, J.E., Nauwkeurig bepalen van grondverzet op basis van CAD ontgravingsprofielen en GIS, een haalbaarheidsstudie, Rapport aan de Bouwdienst van Rijkswaterstaat, Delft 2001, 23 p.
11. GISt Report No. 11, Geo DBMS, De basis van GIS-toepassingen, KvAG/AGGN Themamiddag, 14 november 2001, J. Flim (eds.), Delft 2001, 37 p.
12. GISt Report No. 12, Vries, M.E. de, T.P.M. Tijssen, J.E. Stoter, C.W. Quak and P.J.M. van Oosterom, The second GML prototype of the new TOP10vector object model, Report for the Topographic Service, Delft 2002, Part 1, Main text, 63 p. and Part 2, Appendices B and C, 85 p.
13. GISt Report No. 13, Vries, M.E. de, T.P.M. Tijssen en P.J.M. van Oosterom, Comparing the storage of Shell data in Oracle spatial and in Oracle/ArcSDE compressed binary format, Delft 2002, .72 p. (Confidential)
14. GISt Report No. 14, Stoter, J.E., 3D Cadastre, Progress Report, Report to Concernstaf Kadaster, Delft 2002, 16 p.
15. GISt Report No. 15, Zlatanova, S., Research Project on the Usability of Oracle Spatial within the RWS Organisation, Detailed Project Plan (MD-NR. 3215), Report to Meetkundige Dienst – Rijkswaterstaat, Delft 2002, 13 p.
16. GISt Report No. 16, Verbree, E., Driedimensionale Topografische Terreinmodellering op basis van Tetraëder Netwerken: Top10-3D, Report aan Topografische Dienst Nederland, Delft 2002, 15 p.
17. GISt Report No. 17, Zlatanova, S. Augmented Reality Technology, Report to SURFnet bv, Delft 2002, 72 p.
18. GISt Report No. 18, Vries, M.E. de, Ontsluiting van Geo-informatie via netwerken, Plan van aanpak, Delft 2002, 17p.
19. GISt Report No. 19, Tijssen, T.P.M., Testing Informix DBMS with spatial data from the cadastre, Delft 2002, 62 p.

20. GISt Report No. 20, Oosterom, P.J.M. van, Vision for the next decade of GIS technology, A research agenda for the TU Delft the Netherlands, Delft 2003, 55 p.
21. GISt Report No. 21, Zlatanova, S., T.P.M. Tijssen, P.J.M. van Oosterom and C.W. Quak, Research on usability of Oracle Spatial within the RWS organisation, (AGI-GAG-2003-21), Report to Meetkundige Dienst – Rijkswaterstaat, Delft 2003, 74 p.
22. GISt Report No. 22, Verbree, E., Kartografische hoogtevoorstelling TOP10vector, Report aan Topografische Dienst Nederland, Delft 2003, 28 p.
23. GISt Report No. 23, Tijssen, T.P.M., M.E. de Vries and P.J.M. van Oosterom, Comparing the storage of Shell data in Oracle SDO_Geometry version 9i and version 10g Beta 2 (in the context of ArcGIS 8.3), Delft 2003, 20 p. (Confidential)
24. GISt Report No. 24, Stoter, J.E., 3D aspects of property transactions: Comparison of registration of 3D properties in the Netherlands and Denmark, Report on the short-term scientific mission in the CIST – G9 framework at the Department of Development and Planning, Center of 3D geo-information, Aalborg, Denmark, Delft 2003, 22 p.
25. GISt Report No. 25, Verbree, E., Comparison Gridding with ArcGIS 8.2 versus CPS/3, Report to Shell International Exploration and Production B.V., Delft 2004, 14 p. (confidential).
26. GISt Report No. 26, Penninga, F., Oracle 10g Topology, Testing Oracle 10g Topology with cadastral data, Delft 2004, 48 p.
27. GISt Report No. 27, Penninga, F., 3D Topography, Realization of a three dimensional topographic terrain representation in a feature-based integrated TIN/TEN model, Delft 2004, 27 p.
28. GISt Report No. 28, Penninga, F., Kartografische hoogtevoorstelling binnen TOP10NL, Inventarisatie mogelijkheden op basis van TOP10NL uitgebreid met een Digitaal Hoogtemodel, Delft 2004, 29 p.
29. GISt Report No. 29, Verbree, E. en S.Zlatanova, 3D-Modeling with respect to boundary representations within geo-DBMS, Delft 2004, 30 p.
30. GISt Report No. 30, Penninga, F., Introductie van de 3e dimensie in de TOP10NL; Voorstel voor een onderzoekstraject naar het stapsgewijs introduceren van 3D data in de TOP10NL, Delft 2005, 25 p.
31. GISt Report No. 31, P. van Asperen, M. Grothe, S. Zlatanova, M. de Vries, T. Tijssen, P. van Oosterom and A. Kabamba, Specificatie datamodel Beheerkaart Nat, RWS-AGI report/GIST Report, Delft, 2005, 130 p.
32. GISt Report No. 32, E.M. Fendel, Looking back at Gi4DM, Delft 2005, 22 p.
33. GISt Report No. 33, P. van Oosterom, T. Tijssen and F. Penninga, Topology Storage and the Use in the context of consistent data management, Delft 2005, 35 p.
34. GISt Report No. 34, E. Verbree en F. Penninga, RGI 3D Topo - DP 1-1, Inventarisatie huidige toegankelijkheid, gebruik en mogelijke toepassingen 3D topografische informatie en systemen, 3D Topo Report No. RGI-011-01/GISt Report No. 34, Delft 2005, 29 p.
35. GISt Report No. 35, E. Verbree, F. Penninga en S. Zlatanova, Datamodellering en datastructurering voor 3D topografie, 3D Topo Report No. RGI-011-02/GISt Report No. 35, Delft 2005, 44 p.
36. GISt Report No. 36, W. Looijen, M. Uitentuis en P. Bange, RGI-026: LBS-24-7, Tussenrapportage DP-1: Gebruikerswensen LBS onder redactie van E. Verbree en E. Fendel, RGI LBS-026-01/GISt Rapport No. 36, Delft 2005, 21 p.
37. GISt Report No. 37, C. van Strien, W. Looijen, P. Bange, A. Wilcsinszky, J. Steenbruggen en E. Verbree, RGI-026: LBS-24-7, Tussenrapportage DP-2: Inventarisatie geo-informatie en -services onder redactie van E. Verbree en E. Fendel, RGI LBS-026-02/GISt Rapport No. 37, Delft 2005, 21 p.
38. GISt Report No. 38, E. Verbree, S. Zlatanova en E. Wisse, RGI-026: LBS-24-7, Tussenrapportage DP-3: Specifieke wensen en eisen op het gebied van plaatsbepaling, privacy en beeldvorming, onder redactie van E. Verbree en E. Fendel, RGI LBS-026-03/GISt Rapport No. 38, Delft 2005, 15 p.

39. GISt Report No. 39, E. Verbree, E. Fendel, M. Uitentuis, P. Bange, W. Looijen, C. van Strien, E. Wisse en A. Wilcsinszky en E. Verbree, RGI-026: LBS-24-7, Eindrapportage DP-4: Workshop 28-07-2005 Geo-informatie voor politie, brandweer en hulpverlening ter plaatse, RGI LBS-026- 04/GISt Rapport No. 39, Delft 2005, 18 p.
40. GISt Report No. 40, P.J.M. van Oosterom, F. Penninga and M.E. de Vries, Trendrapport GIS, GISt Report No. 40 / RWS Report AGI-2005-GAB-01, Delft, 2005, 48 p.
41. GISt Report No. 41, R. Thompson, Proof of Assertions in the Investigation of the Regular Polytope, GISt Report No. 41 / NRM-ISS090, Delft, 2005, 44 p.
42. GISt Report No. 42, F. Penninga and P. van Oosterom, Kabel- en leidingnetwerken in de kadastrale registratie (in Dutch) GISt Report No. 42, Delft, 2006, 38 p.
43. GISt Report No. 43, F. Penninga and P.J.M. van Oosterom, Editing Features in a TEN-based DBMS approach for 3D Topographic Data Modelling, Technical Report, Delft, 2006, 21 p.
44. GISt Report No. 44, M.E. de Vries, Open source clients voor UMN MapServer: PHP/Mapscript, JavaScript, Flash of Google (in Dutch), Delft, 2007, 13 p.
45. GISt Report No. 45, W. Tegtmeier, Harmonization of geo-information related to the lifecycle of civil engineering objects – with focus on uncertainty and quality of surveyed data and derived real world representations, Delft, 2007, 40 p.
46. GISt Report No. 46, W. Xu, Geo-information and formal semantics for disaster management, Delft, 2007, 31 p.
47. GISt Report No. 47, E. Verbree and E.M. Fendel, GIS technology – Trend Report, Delft, 2007, 30 p.
48. GISt Report No. 48, B.M. Meijers, Variable-Scale Geo-Information, Delft, 2008, 30 p.
49. GISt Report No. 48, Maja Bitenc, Kajsa Dahlberg, Fatih Doner, Bas van Goort, Kai Lin,Yi Yin, Xiaoyu Yuan and Sisi Zlatanova, Utilty Registration, Delft, 2008, 35 p.
50. GISt Report No 50, T.P.M. Tijssen en S. Zlatanova, Oracle Spatial 11g en ArcGIS 9.2 voor het beheer van puntenwolken (Confidential), Delft, 2008, 16 p.
51. GISt Report No. 51, S. Zlatanova, Geo-information for Crisis Management, Delft, 2008, 24 p.
52. GISt Report No. 52, P.J.M. van Oosterom, INSPIRE activiteiten in het jaar 2008 (partly in Dutch), Delft, 2009, 142 p.
53. GISt Report No. 53, P.J.M. van Oosterom with input of and feedback by Rod Thompson and Steve Huch (Department of Environment and Resource Management, Queensland Government), Delft, 2010, 60 p.
54. GISt Report No. 54, A. Dilo and S. Zlatanova, Data modeling for emergency response, Delft, 2010, 74 p.
55. GISt Report No. 55, Liu Liu, 3D indoor " door-to-door" navigation approach to support first responders in emergency response – PhD Research Proposal, Delft, 2011, 47 p.
56. GISt Report No. 56, Md. Nazmul Alam, Shadow effect on 3D City Modelling for Photovoltaic Cells – PhD Proposal, Delft, 2011, 39 p.
57. GIST Report No. 57, G.A.K. Arroyo Ohori, Realsing the Foundations of a Higher Dimensional GIS: A Study of Higher Dimensional Data Models, Data Structures and Operations – PhD Research Proposal, Delft, 2011, 68 p.
58. GISt Report No. 58, Zhiyong Wang, Integrating Spatio-Temporal Data into Agent-Based Simulation for Emergency Navigation Support – PhD Research Proposal, Delft, 2012, 49 p.
59. GISt Report No. 59, Theo Tijssen, Wilko Quak and Peter van Oosterom, Geo-DBMS als standard bouwsteen voor Rijkswaterstaat (in Dutch), Delft, 2012, 167 p.
60. GISt Report No. 60, Amin Mobasheri, Designing formal semantics of geo-information for disaster response – PhD Research Proposal, Delft, 2012, 61 p.
61. GISt Report No. 61, Simeon Nedkov, Crowdsourced WebGIS for routing applications in disaster management situations, Delft, 2012, 31 p.

62. GISt Report No. 62, Filip Biljecki, The concept of level of detail in 3D city models – PhD Research Proposal, Delft, 2013, 58 p.
63. GISt Report No. 63, Theo Tijssen & Wilko Quak, GISt activiteiten voor het GeoValley project – Projectnummer: GBP / 21F.005, Delft, 2013, 39 p.
64. GISt Report No. 64, Radan Šuba, Content of Variable-scale Maps – PhD Proposal, Delft, 2013, 36 p.
65. GISt Report No. 65, Ravi Peters, Feature aware Digital Surface Model analysis and generalization based on the 3D Medial Axis Transform, - PhD Research Proposal, Delft 2014, 55 p.
66. GISt Report No. 66, Sisi Zlatanova, Liu Liu, George Sithole, Junqiao Zhao and Filippo Mortani, Space subdivision for indoor applications, Delft, 2014, 38 p.
67. GISt Report No. 67, Sisi Zlatanova, Jakob Beetz, Joris Goos, Albert Mulder, Anne-Jan Boersma, Hans Schevers, Marian de Vries and Tarun Ghawana, Spatial Infrastructure for the Port of Rotterdam, Delft, 2014, 35 p.
68. GISt Report No. 68, Edward Verbree en Peter van Oosterom, Exploratieve Puntenwolken, Delft, 2015, 60 p.
69. GISt Report No. 69, Wilko Quak, Storage of spatial properties of NWB-vaarwegen in Neo4j, Delft, 2016, 13 p.
70. GISt Report No. 70, Edward Verbree and Peter van Oosterom, Standaardisatie van Puntenwolken, Delft, 2016, 66 p.
71. GISt Report No. 71, Martijn Meijers, Peter van Oosterom and Wilko Quak, Management of AIS messages in a Geo-DBMS, Delft, 2018, 33 p.

OTB – Research for the built environment
Faculty of Architecture and the Built Environment, TU Delft
Julianalaan 134, 2628 BL Delft
Postbus 5043, 2600 GA Delft

Phone: +31 (0)15 278 30 05
E-mail: OTB-bk@tudelft.nl
www.otb.bk.tudelft.nl