APPLICATION OF THE SPATIAL DATABASE FOR SHORELINE CHANGE ANALYSIS AND VISUALISATION: EXAMPLE FROM THE WESTERN POLISH COAST, SOUTHERN BALTIC SEA

Robert Kostecki

Institute of Geoecology and Geoinformation, Adam Mickiewicz University in Poznań, Poland

Manuscript received: September 25, 2017
Revised version: August 3, 2018

Kostecki R., 2018. Application of the spatial database for shoreline change analysis and visualisation: example from the western Polish coast, southern Baltic Sea. *Quaestiones Geographicae* 37(3), Bogucki Wydawnictwo Naukowe, Poznań, pp. 25–34. 11 figs.

Abstract: The main aim of the study was to introduce a spatial database application for the estimation of changes in shoreline position. The open-source PostgreSQL database system with the PostGIS spatial extension was used as the data store for digitalised shorelines. The solution to calculations of the shoreline changes was based on the functions written in the PL/SQL language and geospatial functions provided by the PostGIS extension. The traditional *baseline and transects* method was used to quantify the distances and rate of shoreline movement. Outputs of the calculations were stored in the database table and simply visualised using graphical functions in the R software environment or in GIS Desktop software. The advantage of presented method is the application of SQL language in the analysis of the relation between the geometry of shorelines stored in the database table, which, compared to other similar solutions, gives the user fully open, simple analytical code and enable selecting custom parameters of analysis, modifying code and performing additional calculations.

Key words: shoreline change analysis, spatial database, open source GIS software, PostGIS, PostgreSQL

Corresponding author: Robert Kostecki, kostecki@amu.edu.pl

Introduction

The development of geographical information systems (GIS) software programs has increased the availability of a number of computer-aided methods suitable for developing estimations of the shoreline change. Most of the solutions provide ready GIS-tools dedicated to shoreline analysis, while the development of the programming languages, spatial databases, and programming libraries also allows researchers to create their own tools and to perform estimations with full control throughout the code. The main aim of this study is to introduce a new method of estimation of shoreline change and to present an example of its use and the results obtained. The selected shoreline of Polish part of Uznam Island and Wolin Island as case study has been subject of many previous studies (Kostrzewski and Zwoliński 1988, 1995, Kostrzewski et al. 2015, Terefenko et al. 2018), that confirmed distinct cliff retreat in Wolin Island.

The GIS tool presented here enables similar estimations of the rate of shoreline change from multiple historical shorelines as it is calculated by the Digital Shoreline Analysis System (DSAS) software extension of ArcGIS (Thieler et al. 2009) or the Analysing Moving Boundaries Using R (AMBUR) package of the R software environment (Jackson et al. 2012). There are many studies that present examples of the application of the DSAS tool in the detection of shoreline changes in

DE GRUYTER OPEN

© 2018 Author(s)
This is an open access article distributed under
the Creative Commons Attribution-NonCommercial-NoDerivs license

doi: 10.2478/quageo-2018-0023
ISSN 0137-477X, eISSN 2081-6383

different parts of the world (Ahmad and Lakhan 2012, Chaaban et al. 2012, Montreuil and Bullard 2012, Liu et al. 2013, Río et al. 2013, Kolega 2015). The questions in social forums about an open-GIS tool that works similarly to DSAS highlight the need to develop tools and methods that perform estimation of the shoreline change, especially in open-source GIS applications, while allowing users to control and modify the processes of the calculation and that are available without a commercial license. The solution presented is based on Structured Query Language (SQL) and the PostgreSQL database system (Obe and Hsu 2012) with a PostGIS spatial extension.

PostgreSQL is a well-known open source database management system that allows users to store data in a database structure, perform management and analysis using the SQL statements, and create analytical functions. PostGIS is a spatial database extender for PostgreSQL system that has the ability to store spatial data in the tables and perform spatial analysis. The analytical possibilities of PostGIS make the database a powerful GIS tool. The calculations performed were based on the *baseline and transect* method (Dolan et al. 1978), which is often used to estimate shoreline position changes (Dudzińska-Nowak and Furmańczyk 2005, Chaaban et al. 2012, Jackson et al. 2012).

The *baseline and transect* method involves few calculation steps. Firstly the user should prepare a baseline by drawing a reference line onshore or offshore from the analysed shorelines as suggested in the solutions such as DSAS or AMBUR. In this study, the baseline was created during the execution function from the kilometre points assigned to the coast. The second step is to create transect lines perpendicular to the baseline with user-defined distances between lines. The user can create transect lines automatically using the PostgreSQL function introduced in this paper. The next step is to calculate the distance along the transect between the points determined from intersections of the transect with historical shorelines. This method provides an analytical function that makes calculations based on the distances described above, and the results can be stored in the database tables. Other additional calculations and visualisations could be performed in the database server environment using language extensions or by connecting with the R statistical environment. The most important advantages of the presented method are the simplicity and complete control of the whole code by the user. The whole code of the presented solution is based on SQL language and the open software database system and all required SQL statements are presented in this paper. The solution allow user implement own modifications at each stage of the code.

A similar approach to measure changes in shoreline position was performed by Chaaban et al. (2012) using ArcGIS commercial software. The presented attempt does not require a commercial license and installation of additional packages but only the open-source database system with the spatial extension and, optionally, the R statistical environment. The user have access to the whole source code and also can modify the code to suit his needs. The output of the program is the transects table, which stores the geometry of the transects and all results of calculations. Such a solution offers the user a fast and simple visualisation of the results and the option of performing additional calculations. Furthermore, the transect lines are a result of cutting along the shortest lines the envelope containing all analysed shorelines. The envelope is created from the baseline and their offset line. The shape of the baseline is independent from the shape of the analysed shorelines but depends only on the points along the shore prepared by the user, and thereby measurements are not affected by interruption of the shoreline at estuaries and the curved shape of the shoreline.

Method development

The methodology for estimation of shoreline movement is based on the SQL functions executed in the presented example of the spatial database. The code of functions and the execution procedure are detailed below.

Data Sources

The example shoreline comprises 37 km of the western part of the Polish coastline along the Baltic Sea (Fig. 1). The digitalisation of the shoreline was performed in the QGIS open geosciences application. The shorelines were retrieved from topographic map at 1:10,000 scale re-projected from the geodetic spatial reference system 1965/3

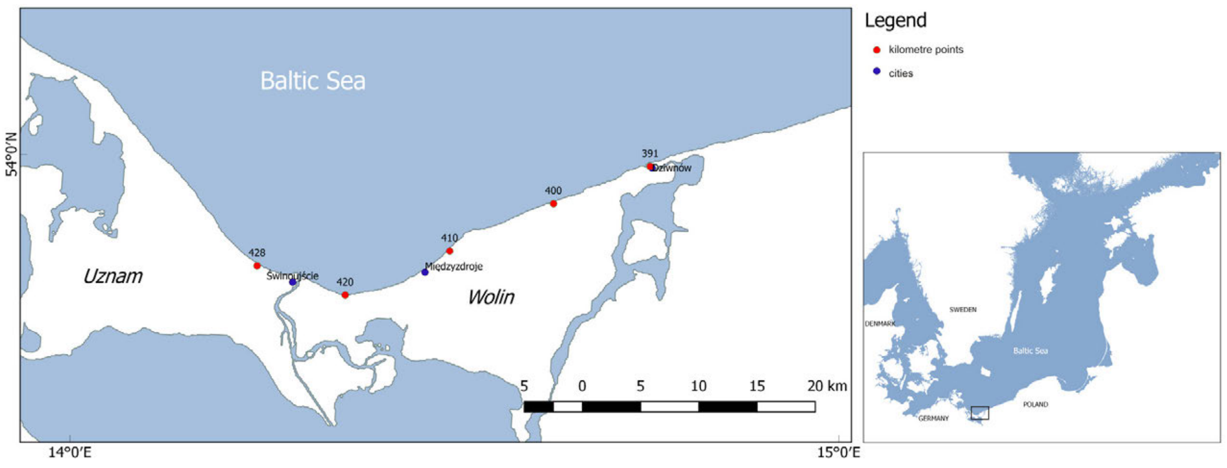


Fig. 1. Location map of tested shoreline.

(SRID: 2173) with actualization in the years 1987 to 1989 and the third orthophoto map was made from aerial photos from 2010. All maps are available at the Polish Geoportal as the Web Map Service (WMS) and are provided by the Polish Head Office of Geodesy and Cartography.

Accuracy of Data

The presented example of shoreline changes analysis was based on the lines retrieved from different data sources of varying quality: topographic maps and orthophotographic images. The dune/cliff foot line in the maps and orthophotographic images was identified according to erosion reference features such as the top edge of a bluff, dune escarpment, or vegetation line (Crowell et al. 2005). Digitalised shorelines from scanned and georeferenced topographic maps at 1:10,000 scale provide a position error of ± 10 m, while digitalisation of orthophoto maps with a pixel size of 0.5 m provides a maximal error of ± 5 m. It should be taken into account that the error margin doubled when comparing the results of shoreline position from different data sources, and in the presented case the total position error was estimated as a maximum of around ± 15 m.

Structure of database

The geospatial data were stored in the PostgreSQL database management system with the spatial extension PostGIS (Obe and Hsu 2015). The spatial extension enables the database to store spatial geometry types and execute spatial analysis with the support of the special functions and SQL statements. The first step in the preparation of the database was the creation of spatial tables suitable for storing the geometry of the shoreline with attributes. The spatial tables were used as the store of the vector lines instead of the ESRI shapefile format method that is often used. Obviously the user can import previously prepared vector data in the shapefile format to the database system. The result of the import will be stored as a spatial table. The shoreline table should be prepared for each time period.

In the presented case, the three tables were prepared to store shorelines digitalised from maps from three time periods and one table to store kilometric points of the shoreline. All the tables were stored in the schema container called *gis*. The tables included three columns: *gid* as the unique identifier (ID) of the object, *type* as the description of the kind of line (*the waterline* or *the footline of the foredune or cliff*), and *geom* as a binary representation of the geometry. The type of geometry was set as *linestring* in the Polish geodetic spatial reference system *1992* (ETRS89/Poland CS92) with spatial reference identifier (SRID) 2180. The supplementary table consisting of kilometric points of the Polish coast digitalised from topographic maps was called *gis.kilometre*. The *gis.kilometre* table contained the following columns: *km* as a unique ID and *geom* as a binary representation of the geometry. The table of kilometric points is the base from which the transects are created, and in cases where the shoreline is not a straight line, the points should be placed along the shoreline at a greater density than one per kilometre. Digitalised vector layers of the shorelines were stored in the separate tables *shoreline_1987*, and *shoreline_2010*. Each table included the geometry of two lines: the waterline and the footline of the foredune or cliff line. The waterline is an unstable boundary between land and sea that is controlled by the wave conditions, while the best independent proxy for shoreline determination is the footline of the foredune or cliff (Dudzińska-Nowak and Furmańczyk 2005, Río et al. 2013). The functions allow the user to select the type of analysed line. All calculations of shoreline changes were based on the footline.

Analytical Procedure and Outputs

The analytical procedure and outputs are presented in a workflow diagram (Fig. 2), while all necessary statements are presented in Figure 3. The first step of the procedure is the creation of a table to store the transect lines and calculation results, called for example *transects_table*. The created table should contain at least the fields

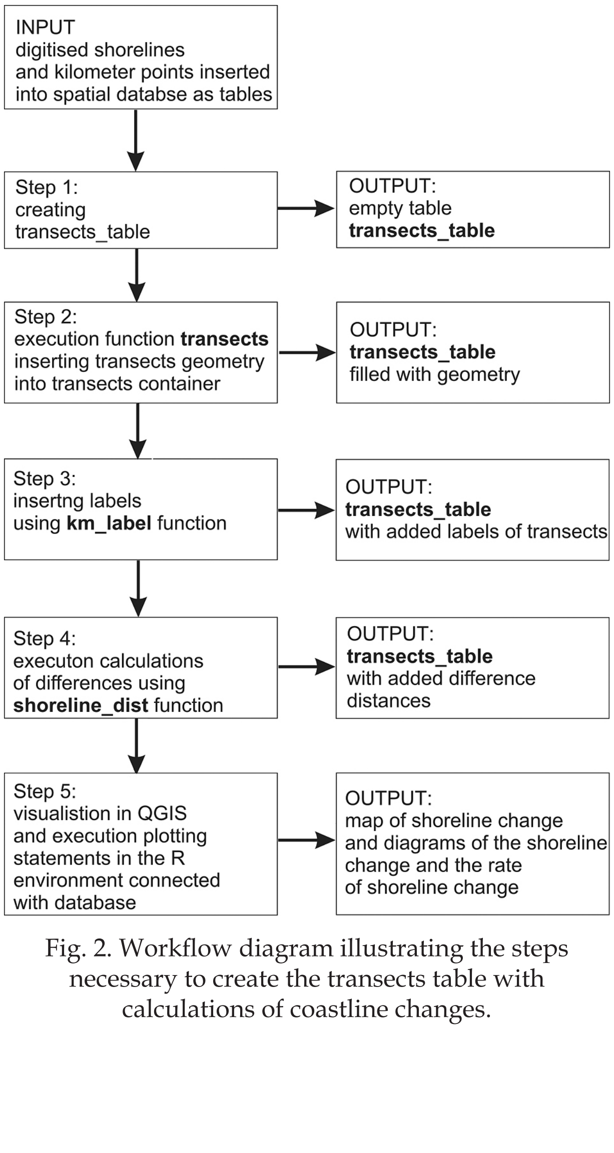


Fig. 2. Workflow diagram illustrating the steps necessary to create the transects table with calculations of coastline changes.

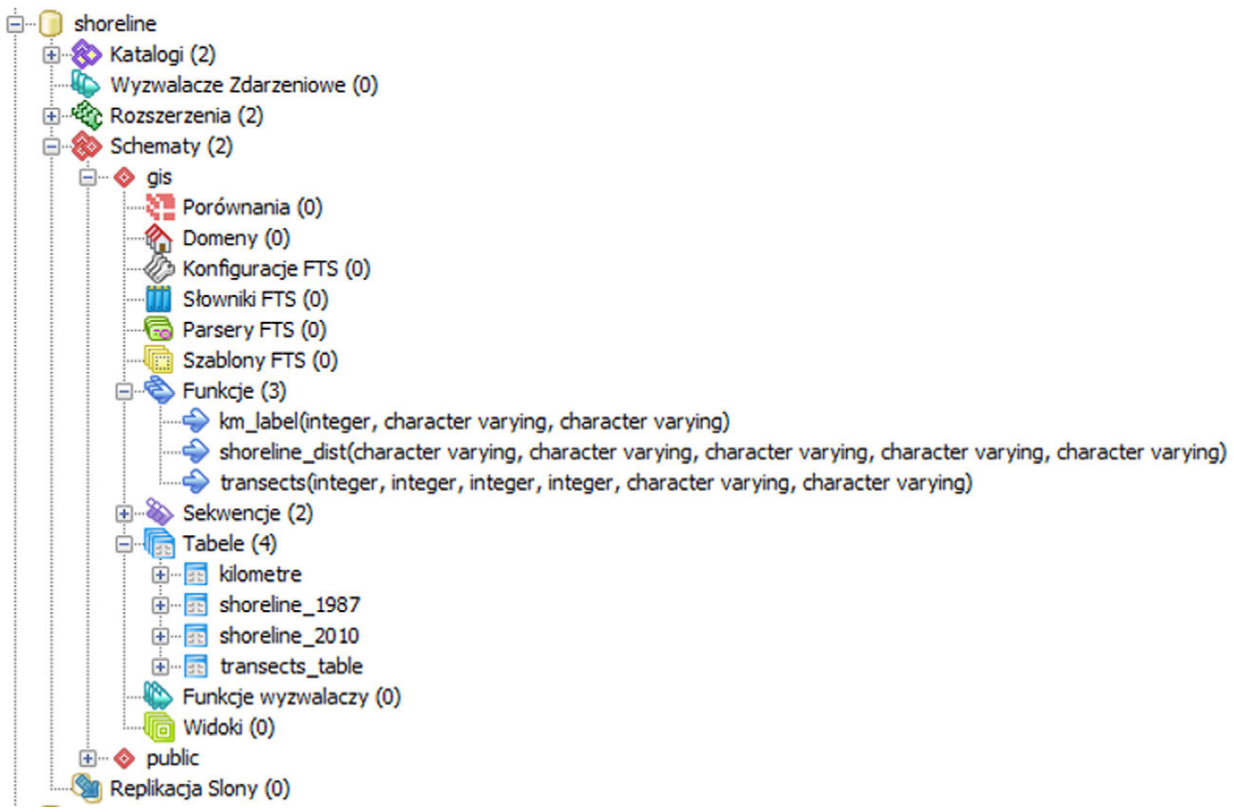
```
--creating the transects table
create table gis.transects_table(
gid serial NOT NULL primary key,
km numeric(5,2),
dist_foot double precision,
geom geometry(linestring,2180)
);
--inserting transect lines
insert into gis.transects_table(gid, geom)
select * from gis.transects(250,10,391,428,’gis’,’kilometre’);
--labelling transects
update gis.transects_table
set km =
(select km from gis.km_label(391,’gis’,’transects_table’)as a
where transects_table.gid =a.gid);
--calculations of differences between selected shorelines
update gis.transects_table
set dist_foot=
(select dist from gis.shoreline_dist(‘gis’,’transects_table’,’shore-
line_1987’,’shoreline_2010’,
‘the footline%’)as a
where transects_table.gid =a.gid)
```

Fig. 3. Example of SQL execution statements necessary to create the geometry of transects and perform transect calculations.

prepared for the unique ID and geometry of objects. The following steps create the geometry of the transects by inserting statement and function *transects* with arguments presented in Figure 3 and adding labels to create transect objects using the *km_label* function in an update statement. An example of the transect lines created is presented in Figure 5. The transects table can also store the results of calculations in the separate fields. In the presented example, the *dist_foot* double precision field was prepared, ready to store the calculated distances between footlines. The calculations were performed by the function *shoreline_dist* and the results were inserted using the update statement (see Fig. 3).

Description of the functions

The presented functions were written in the PL/SQL procedural language. Before starting the computations it was necessary to load into the database the following functions: *transects* (all the names of presented functions and tables used in the SQL statements should contain the prefix of the schema name), which created the geometry transects; *coast_diff*, which performed the shoreline difference calculations; and *km_label*, a transect-labelling function. The example of database elements ready to calculations is presented in Figure 4. All required functions and statements are presented in Figures 5–8.



```
shoreline
  Katalogi (2)
  Wyzwalacze Zdarzeniowe (0)
  Rozszerzenia (2)
  Schematy (2)
    gis
      Porównania (0)
      Domeny (0)
      Konfiguracje FTS (0)
      Słowniki FTS (0)
      Parsery FTS (0)
      Szablony FTS (0)
      Funkcje (3)
        km_label(integer, character varying, character varying)
        shoreline_dist(character varying, character varying, character varying, character varying, character varying)
        transects(integer, integer, integer, integer, character varying, character varying)
      Sekwencje (2)
      Tabele (4)
        kilometre
        shoreline_1987
        shoreline_2010
        transects_table
      Funkcje wyzwalaczy (0)
      Widoki (0)
    public
  Replikacja Slony (0)
```

Fig. 4. Preview of example PostgreSQL database structure in window of the client application pgAdmin III with all necessary tables and functions to made calculations.

Functions transects and labelling

The input arguments of the *transects* function (Fig. 6) are the *length* as an integer value of the transect length (the transect lines should intersect all analysed shorelines); *tinterval* as an integer value of the interval between the created transects; *from_km* as an integer value of the first kilometre of analysed coast (this value requires the *km* field in the *kilometre* table); *to_km* as an integer value of the final kilometre of analysed coast; names of schema containers (*sche*); and the table

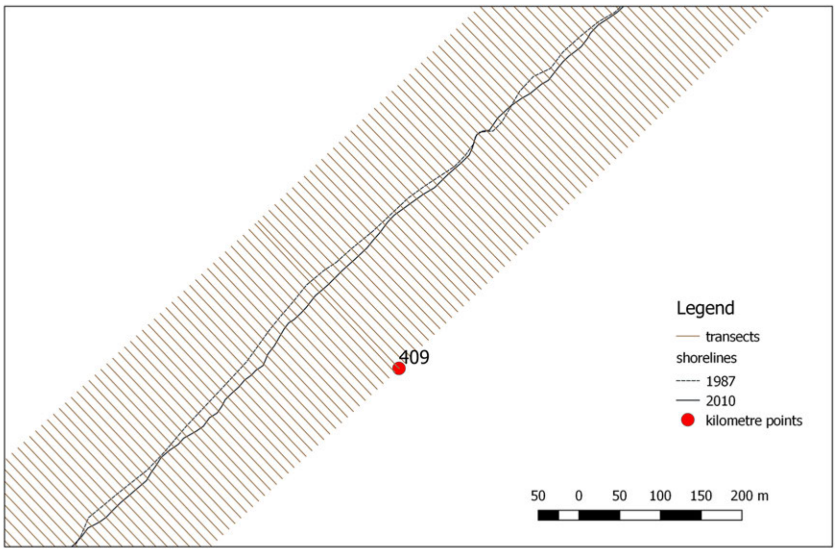


Fig. 5. Example for erosional part of shoreline with resultant transect lines at 10-m spacing intersected and digitalised historic shorelines.

```
CREATE OR REPLACE FUNCTION gis.transects(
            in length integer,
            in tinterval integer,
            in from_km integer,
            in to_km integer,
            in sche varchar,
            in tbl varchar,
            out gid integer,
            out geom geometry
            )
 RETURNS SETOF record AS
$$
DECLARE
            rec record;
            sql text;
BEGIN
gid := 0;
sql :=
‘with a as
(select ((st_DumpPoints(St_Segmentize(St_Makeline(p.ge-
om),’||tinterval||’))).geom)
as point_track
from
(select km, geom as geom from ‘||quote_
ident(sche)||’.’||quote_ident(tbl)||’
where km>=’||from_km||
‘and km<=’||to_km ||’ order by km desc)p),
--creating offset line based on reference line
b as
(
select ST_OffsetCurve(St_Makeline(p.geom),’||length||’,
’’quad_segs=8 join=round’’)
as track
from
(select km, geom as geom from ‘||quote_
ident(sche)||’.’||quote_ident(tbl)||’
where km>=’||from_km||
‘and km<=’||to_km ||’ order by km desc)p
)
--creating the transects lines
select st_makeline(a.point_track,st_closestpoint(b.track,a.
point_track))as geom
from a, b where st_dwithin(b.track,a.point_
track,’||length+50||’)’;
for rec in execute sql
LOOP
            gid := gid + 1;
            geom := rec.geom;
            RETURN NEXT;
END LOOP;
RETURN;
END;
$$
LANGUAGE ‘plpgsql’ STABLE;
```

Fig. 6. PL/SQL source code for function necessary to creating geometry of the transects along the analysed shoreline (detailed description in the text).

(*tbl*) with digitalised kilometre points along the analysed coast. The output arguments consisted of *gid* as a unique ID and *geom* as the geometry of the transect object. The *transects* function works in the following sequence of the PostGIS functions: *ST_Makeline*, *ST_Segmentize*, and *ST_DumpPoints*. The function *ST_Makeline* creates the baseline along the shoreline between kilometre points,

```
CREATE OR REPLACE FUNCTION gis.km_label(
            in start_km integer,
            in sche varchar,
            in tbl varchar,
            out km numeric(4,1),
            out gid integer
            )
RETURNS SETOF record AS
$$
DECLARE
            rec record;
            sql text;
BEGIN
km := start_km;
sql :=
‘select gid, coalesce(st_distance(st_startpoint(lag(‘||quote_
ident(tbl)||’.geom)
over(order by gid desc)),st_startpoint(‘||quote_ident(tbl)||’.
geom)),0)as dist
 from ‘||quote_ident(sche)||’.’||quote_ident(tbl);
for rec in execute sql
LOOP
            gid := rec.gid;
            km := km+rec.dist*0.001;
            RETURN NEXT;
END LOOP;
RETURN;
end;
$$
LANGUAGE ‘plpgsql’ STABLE;
```

Fig. 7. PL/SQL source code for function labelling transects according to kilometre names.

while the function *ST_Segmentize* divides the baseline into segments at intervals selected by the user. The segments were transformed into points by the *ST_DumpPoints* function called *point_track*. The reference line from the kilometre points should not intersect with the analysed shorelines. The next step of the function is to make an offset line called *track* using *ST_OffsetCurve* and *ST_ Makeline* and a value *length*, which is the distance of the offset. The *track* and *point_track* point lines should bound the area that contains all analysed shorelines. The final step of the procedure is to make the transect line geometry using *ST_Closest* point and the condition of searching *ST_DWithin* with the selected distance *length* enlarged by 50 m.

The example of the resultant transect lines is presented in Figure 5. The resultant transects are labelled by kilometre names using the function *km_label*. The supplemental function *km_label* made labels for transects according to kilometre markings along the shoreline (Fig. 7). The main task of the function is to calculate distances between transects and assign kilometre labels according to the real kilometre mark supplied by the function *start_km*. Transects were labelled in ascending order from the right side, as in the case of the Polish coastline. The code of function is flexible and allow select custom parameters of calculation such as density of transects (*tinterval*), the length of transects lines (*length*), analysed segment of coast (*from_km*, *to_km*) or method of labelling (function *km_label*). In presented example selected 10 m as density of transects, 250 m as length of transects and segment of coast between kilometre points 391 and 428.

Function for calculating changes in shoreline position

The function *shoreline_dist* (Fig. 8) requires the following input arguments: *sche* as the name of the schema container; *transects_tbl* as the name of the transect table; *l1_tbl* as the name of the table of older shorelines; *l2_tbl* as the name of the table of younger shorelines; and *type* as the description of the kind of shoreline. The output argument of the function consists of an integer variable *gid* as an ID for the transect and the double precision variable *dist* as a result of the computations. The first step of the function is to determine the point geometry from the intersection of the transect line with the older shoreline, called variable *l1*, and the younger shoreline, called variable *l2*. The next step is to determine the geometry of the start points of the transects, called *tpoint*, which allows one to perform calculations of the distances. The measured distances between the points *tpoint*, *l1*, and *l2* were stored as the double precision variables *dist1* and *dist2*. The final results were calculated as an algebraic subtraction of these distances and those stored for the recorded variable *dist*. If the shoreline moves into the sea, the values of subtraction calculation are positive, while if the shoreline moves inland, the values are negative.

Case study results

The best way to visualise the results is through the R statistical environment (Kabacoff 2011) connected to the PostgreSQL database. The database driver and standard R graphical functions perform data visualisation techniques loaded into the R data frame from a database table. The measurements of the rate of shoreline changes per year between different shoreline positions

```
CREATE OR REPLACE FUNCTION gis.shoreline_dist(
            in sche varchar,
            in transects_tbl varchar,
            in l1_tbl varchar,
            in l2_tbl varchar,
            in type varchar,
            out gid integer,
            out dist double precision
            )
 RETURNS SETOF record AS
$$
DECLARE
            rec record;
            sql text;
BEGIN
sql :=
‘with
--intersection of the shoreline1 with transects
l1 as(
select ‘||quote_ident(transects_tbl)||’.gid,st_intersection(‘
||quote_ident(l1_tbl)||’.geom,’||quote_ident(transects_
tbl)||’.geom)as geom1
from
‘||quote_ident(sche)||’.’||quote_ident(transects_tbl)||’,
‘||quote_ident(sche)||’.’||quote_ident(l1_tbl)||’
where ‘||quote_ident(l1_tbl)||’.typ like ‘’’||type||’’’and
st_intersects(‘
||quote_ident(l1_tbl)||’.geom,’||quote_ident(transects_
tbl)||’.geom)),’
--intersction of the shoreline2 with transcets
‘l2 as(
select ‘||quote_ident(transects_tbl)||’.gid,st_intersection(‘
||quote_ident(l2_tbl)||’.geom,’||quote_ident(transects_
tbl)||’.geom)as geom2
from
‘||quote_ident(sche)||’.’||quote_ident(transects_tbl)||’,
‘||quote_ident(sche)||’.’||quote_ident(l2_tbl)||’ where
‘||quote_ident(l2_tbl)||
‘.typ like ‘’’||type||’’’ and st_intersects(‘
||quote_ident(l2_tbl)||’.geom,’||quote_ident(transects_
tbl)||’.geom)),’
--startpoints of the transects
‘tpoint as(
select ‘||quote_ident(transects_tbl)||’.gid, st_startpoint(geom)
as geom_tp
from ‘||quote_ident(sche)||’.’||quote_ident(transects_
tbl)||’),
--distance from startpoints to point of the shoreline1
dist1 as(
select l1.gid, min(st_distance(geom_tp,geom1))as dist
from
l1, tpoint where tpoint.gid=l1.gid
group by l1.gid),’
--distance from startpoints to point of the shoreline2
‘dist2 as(
select l2.gid, min(st_distance(geom_tp,geom2))as dist
from
l2, tpoint where tpoint.gid=l2.gid
group by l2.gid)’
--result as difference of the distances
‘select dist1.gid as gid, (dist2.dist-dist1.dist) as dist
from
dist1, dist2
where dist1.gid=dist2.gid’;
for rec in execute sql
LOOP
            gid := rec.gid;
            dist := rec.dist;
            RETURN NEXT;
END LOOP;
RETURN;
end;
$$
LANGUAGE ‘plpgsql’ STABLE;
```

Fig. 8. PL/SQL source code for function calculating distances between analysed shorelines.

```
library("DBI")
library("RPostgreSQL")
drv <- dbDriver("PostgreSQL")
con <- dbConnect(drv, dbname = "mydb", user="postgres")
df <- dbGetQuery(con, statement = paste("select km, dist_foot
from gis.transects_table order by km desc"))
windows(width=12, height=7)
old.par <- par(mfrow=c(2, 1))
barplot((df$dist_foot), ylab ='shoreline change [m]', names.
arg=as.integer(df$km))
segments(0,15,4500,15, col ="grey")
segments(0,-15,4500,-15, col ="grey")
barplot((df$dist_foot/23), ylab ='rate of change [m/yr]',
xlab='Coastline kilometre', names.arg=as.integer(df$km))
segments(0,15/23,4500,15/23, col ="grey")
segments(0,-15/23,4500,-15/23, col ="grey")
```

Fig. 9. R source code statements executed in the R statistical environment necessary to create diagram of the rate of shoreline change.

were made using the end-point rate method (Dolan et al. 1991). The necessary statements executed in the R environment are presented in Figure 9. Graphical plots of the results retrieved from the database table are presented in Figure 10 as plot of change during analysed period with error margin of ±15 m and rate of annual shoreline change with error margin of ±0.65 m.

Another method of visualisation is presenting of shoreline evolution at the map as coloured transects styled according value of determined change during observed period (Fig. 11). The colours represents intervals of adopted classification: strong advance – above 50 m, advance from 15 m to 50 m, fluctuation of advance and retreat

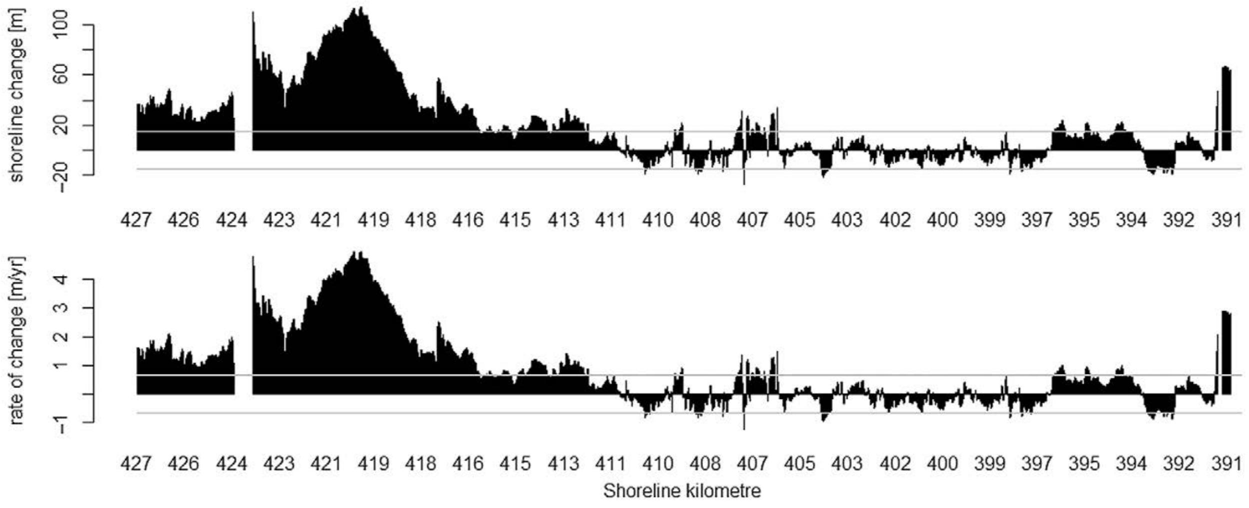


Fig. 10. Example of diagram of shoreline change during analysed period (grey lines: error margin ±15 m) and rate of annual shoreline change (grey lines: error margin ±0.65 m) in the study area created by the R statements on the basis of results stored in the table *transects_table*.

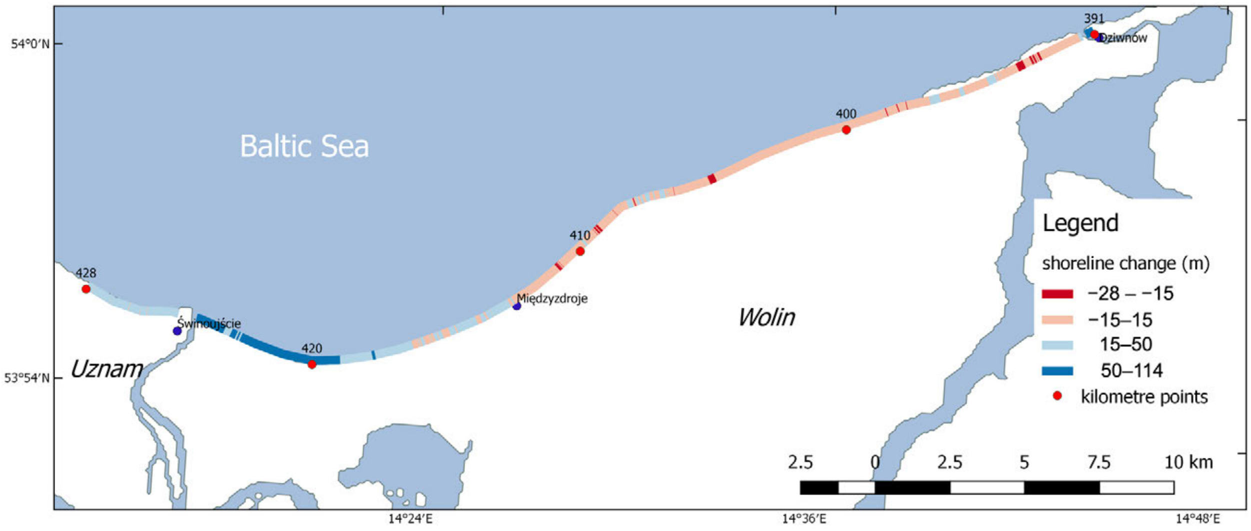


Fig. 11. Example of the map presenting results of classified shoreline change measurements made in the QGIS environment connected with data from PostGIS database.

in error margin from –15 m to 15 m and retreat below –15 m. The westernmost part of analysed shoreline between 428 and 412 km experienced distinct advance. This section is characterised as a typical foredune coast, with high accretion rates from 1 to 4 m a^{-1}. The position of the breakwater of the Świnoujście harbour (425–424 km) is visible as a gap in the plot (Fig. 10) and displays lower accumulation rates than the rest of the area. The eastern part of the breakwater (424–423 km) is the location of the gas terminal built in 2006 and confirmed distinct increase of accumulation processes in this part of the coast. The strong accumulation in this section of the shoreline was confirmed in previous investigations for a longer analysed period of 58 years (Dudzińska-Nowak and Furmańczyk 2005). The shoreline between 391 and 412 km is the cliff coast of Wolin Island and revealed slight retreat of shoreline position particularly between 411 and 408 km and between 405 and 397 km. Some interesting characteristic of this section include the high accretion of shoreline at the mouth of the Dziwna Inlet at 391 km, where the distinct advance of the shore was estimated at 3 m a^{-1}.

Conclusions

The presented study has introduced a new solution for calculations of shoreline change based on the database system PostgreSQL and using spatial functions PostGIS and SQL. The presented methodology was tested on the Polish western coast of the Baltic Sea. Application of the spatial database system to the estimation of shoreline changes highlights the great potential and flexibility of such tools in geospatial analyses, which were previously underestimated. The most important advantages of the presented solution are the database storage method, the open source nature of the applied applications, and full control of the analysis throughout the SQL code. Thanks to this, the user can define custom values of parameters of analysis such as the position and geometry of transects or type of analysed line. In the presented solution nothing is a *black box*, that enable user to modify the code and perform custom calculations. Analyses of shoreline changes in the part of the coast considered as an example provide information about shoreline advance and retreat.

Acknowledgments

The study was financed by the Adam Mickiewicz University in Poznań. The author is grateful to the reviewers for constructive comments.

References

Ahmad S.R., Lakhan V.C., 2012. GIS-Based Analysis and Modeling of Coastline Advance and Retreat Along the Coast of Guyana. *Marine Geodesy* 35: 1–15. DOI:10.1080/01490419.2011.637851.

Chaaban F., Darwishe H., Battiau-Quency Y., Louche B., Masson E., Khattabi J. El, Carlier E., 2012. Using ArcGIS® Modelbuilder and Aerial Photographs to Measure Coastline Retreat and Advance: North of France. *Journal of Coastal Research* 28: 1567–1579. DOI:10.2112/JCOASTRES-D-11-00054.1.

Crowell M., Leatherman S.P., Douglas B.C., 2005. Erosion: historical analysis and forecasting. In: M.L. Schwartz (ed.), *The Encyclopedia of Coastal Science*. Springer, Dordrecht: 428–432.

Dolan R., Hayden B., Heywood J., 1978. A new photogrammetric method for determining shoreline erosion. *Coastal Engineering* 2: 21–39. DOI:10.1016/0378-3839(78)90003-0.

Dolan R., Fenster M., Holme, S., 1991. Temporal Analysis of Shoreline Recession and Accretion. *Journal of Coastal Research* 7: 723–744.

Dudzińska-Nowak J., Furmańczyk K., 2005. Zaminy położenia linii brzegowej Zatoki Pomorskiej (latach 1938–1996). In: R.K. Borówka, S. Musielak (eds.), Środowisko Przyrodnicze Wybrzeży Zatoki Pomorskiej I Zalewu Szczecińskiego. In Plus Oficyna, Szczecin: 72–78.

Jackson C.W., Alexander C.R., Bush, D.M., 2012. Application of the AMBUR R package for spatio-temporal analysis of shoreline change: Jekyll Island, Georgia, USA. *Computers and Geosciences* 41: 199–207. DOI:10.1016/j.cageo.2011.08.009.

Kabacoff R.I., 2011. *R in Action Data analysis and graphics with R*. Manning Publications Co., New York.

Kolega N., 2015. Coastline changes on the Slovenian coast between 1954 and 2010. *Acta Geographica Slovenica* 55–2: 205–221.

Kostrzewski A., Zwoliński Z., 1988. Morphodynamics of the cliffed coast, Wolin Island. *Geographica Polonica* 55: 69–81.

Kostrzewski A., Zwoliński Z., 1995. Present-day morphodynamics of the cliff coasts of Wolin Island. *Journal of Coastal Research* SI: 22: 293–303.

Kostrzewski A., Zwoliński Z., Winowski M., Tylkowski J., Samołyk M., 2015. Cliff top recession rate and cliff hazards for the sea coast of wolin Island (Southern Baltic). *Baltica* 28: 109–120. DOI:10.5200/baltica.2015.28.10.

Liu Y., Huang H., Qiu Z., Fan J., 2013. Detecting coastline change from satellite images based on beach slope estimation in a tidal flat. *International Journal of Applied Earth Observation and Geoinformation* 23: 165–176. DOI:10.1016/j.jag.2012.12.005.

Montreuil A.L., Bullard J.E., 2012. A 150-year record of coastline dynamics within a sediment cell: Eastern England. *Geomorphology* 179: 168–185. DOI:10.1016/j.geomorph.2012.08.008.

Obe R.O., Hsu L.S., 2012. *PostgreSQL:Up and Running*. O'Reilly, Sebastopol.

Obe R.O., Hsu L.S., 2015. *PostGIS in Action, Second Edition*. Manning Publications, Shelter Island.

Río L. Del, Gracia F.J., Benavente J., 2013. Geomorphology Shoreline change patterns in sandy coasts . A case study in SW Spain. *Geomorphology* 196: 252–266. DOI:10.1016/j.geomorph.2012.07.027.

Terefenko P., Giza A., Paprotny D., Kubick, A., Winowski M., 2018. Cliff Retreat Induced by Series of Storms at Międzyzdroje (Poland). *Journal of Coastal Research* SI: 85: 181–185.

Thieler E.R., Himmelstoss E.A., Zichichi J.L., Ergul A., 2009. Digital Shoreline Analysis System (DSAS) version 4.0 – An ArcGIS extension for calculating shoreline change. In: *U.S. Geological Survey Open-File Report 2008*: 1278.