Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings

Volume 15 *Seoul, South Korea* Article 50

2015

Performance Analysis of MongoDB Vs. PostGIS/ PostGreSQL Databases For Line Intersection and Point Containment Spatial Queries

sarthak agarwal
International Institute of Information Technology Hyderabad Gachibowli, Hyderabad, India

KS Rajan
International Institute of Information Technology Hyderabad Gachibowli, Hyderabad, India

AT1-03

Performance Analysis of MongoDB Vs. PostGIS/PostGreSQL Databases For Line Intersection and Point Containment Spatial Queries

Sarthak Agarwal[1] **and KS Rajan**[2]

[1]Lab for Spatial Informatics, International Institute of Information Technology
Gachibowli, Hyderabad 500032, India
Email: sarthak.a@research.iiit.ac.in
[2]Lab for Spatial Informatics, International Institute of Information Technology
Gachibowli, Hyderabad 500032, India
Email: rajan@iiit.ac.in

ABSTRACT

Relational databases have been around for a long time and Spatial databases have exploited this feature for close to two decades. The recent past has seen the development of NoSQL non-relational databases, which are now being adopted for spatial object storage and handling too. And this is gaining ground in the context of increased shift towards GeoSpatial Web Services on both the Web and mobile platforms especially in the user-centric services, where there is a need to improve the query response time. While SQL databases face scalability and agility challenges and fail to take the advantage of the cheap memory and processing power available these days, NoSQL databases can handle the rise in the data storage and frequency at which it is accessed and processed - which are essential features needed in geospatial scenarios, which do not deal with a fixed schema(geometry) and fixed data size.

This paper attempts to evaluate the performance of an existing NoSQL database 'MongoDB' with its inbuilt spatial functions with that of a SQL database with spatial extension 'PostGIS' for two primitive spatial problems - LineIntersection and Point Containment problem, across a range of datasets, with varying features counts. For LineIntersection function, the dataset consisted of two independent layers of horizontal lines and vertical lines with incremental lengths and their size varied from ten lines to million lines in each layer and another dataset with two layers, one of random lines of variable size and shape and another layer of a single line which is intersecting many lines of layer1. For Point Containment problem, the dataset consists of two layers, one of polygons in space of different shape and size and another layer of random points in the space, some inside the polygons and some outside.

All the data in the analysis was processed In-memory and no secondary memory was used. Initial results suggest that MongoDB performs better by an average factor of 25x for Line Intersection Problem and 10x for Point Containment Problem which increases exponentially as the data size increases in both indexed and non-indexed operations. Given these results NoSQL databases may be better suited for simultaneous multiple-user query systems including Web-GIS and mobile-GIS. Further studies are required to understand the full potential of NoSQL databases across various geometries and spatial query types.

1. INTRODUCTION

Traditionally Databases were designed to structure and organize any form of data. But as the database size increased and to optimize databases for Geo Spatial domain we use Spatial Databases. Satellite images are one prominent example of spatial databases. To extract spatial information from a satellite image it has to be processed with respect to a spatial frame of reference. But satellites are not the only type of spatial databases. Maps are also stored in spatial database.

Like other database systems Spatial databases have also relied on the Relational databases to handle and manage spatial objects and their associated attribute information. They have been of great value in cases where we have a defined structure of our schema, including geometrical characteristics of the spatial objects. While these have been able to

store and manage very large datasets, their inherent need to pre-define its structure, database schema, makes it difficult to remodel or change it as per the evolving need, especially in the context of user-driven data collation and collaboration. In addition, in many spatial applications we do not always have a fixed schema, there can be many geometries with different shape and the requirements evolve as the data size increases depending on the case scenario where relational Databases can limit the potential use or design of the solution.

While NoSQL is an umbrella term for a loosely defined class of non-relational datastores and best described as 'Not only SQL'. Examples are Google's BigTable and Amazon's Dynamo. The recent past has seen the development of NoSQL non-relational databases, which are now being adopted for spatial object storage and handling too. And this is gaining ground in the context of increased shift towards GeoSpatial Web Services on both the Web and mobile platforms especially in the user-centric services, where there is a need to improve the query response time, these databases can handle the rise in the data storage and frequency at which it is accessed and processed - which are essential features needed in geospatial scenarios, which do not deal with a fixed schema(geometry) and fixed data size.

Relational databases are not designed to function with distributed systems; combining multiple tables across different computers is difficult and gives mostly a decrease in performance while NoSQL are distributed databases which can be spread over multiple servers and queried simultaneously. Relational databases are good for structured data. For example, sales figures fit well in organized tables. Unstructured data, such as points and lines over a network are not very suitable for a RDBMS, while NoSQL database systems are schema less database systems where we can append any number of geometry columns in a table and store multiple geometries within a same column in a table which is very essential property with respect to spatial context.

NoSQL data stores may provide advantages over relational databases, but generally lack the robustness of relational databases for those advantages. The aim of this paper is to discover some advantages of a selected NoSQL data store as compared to a traditional relational database when storing and querying spatial vector data.

This paper attempt to evaluate the performance of an existing NoSQL database and SQL database with respect to LineIntersection problem and Point Containment problem across a range of datasets. We here are interested in observing which data model performs better with these problem in terms of time complexity without accounting for indexing of the data.

LineIntersection query is the most popular spatial query in which we report the Intersection between two spatial geometries. Geometries can be anything from a point to a polygon. Line Intersection helps in routing as well as mapping various geometries. Suppose we have to make a route from point A to point B ,then from every we point we will find out the lines intersecting it to create a graph using this technique and apply our routing algorithm on that graph.

Like Intersection, Point containment problem is also a very important and popular spatial query problem which reports whether a given geometry lies inside another geometry or not. This problem helps in modeling and analyzing spatial data over a area. For instance, we want to report how the number of houses have changed over a time in a city by analyzing the spatial data of that city. We can count the number of points(which represents each house in this case) in the polygon(city) for all the years using point within a polygon problem.

2. REVIEW OF SPATIAL DATABASES

Similar to traditional databases, Spatial databases were also written initially in RDBMS but as technology evolved they are being now adapted to NoSQL .

2.1 RDBMS(PostGIS / PostgreSQL)

PostgreSQL is a powerful, object-relational database management system (ORDBMS). It is released under a BSD-style license and is thus free and open source software. PostGIS adds support for geographic objects to the PostgreSQL object-relational database. The interface language of the PostgreSQL database is the standard SQL, which allows for inserts, updates and queries of data stored in relational tables. Data operations for PostgreSQL are created in SQL.

For several decades, the most common data storage model for geographic data and associated attribute data has been the relational data model.

The functions in PostGIS can be put into following 5 categories.

- Management: Functions that manage information about spatial tables and PostGIS administration.
- Conversion: Functions that convert between geometries and external data formats.
- Retrieval: Functions that retrieve properties and measurements of a geometry.
- Comparison: Functions that compare two geometries with respect to their spatial relation.
- Generation: Functions that generate new geometries from others

Management depends on other features of a database. Conversion and Retrieval are relatively simple and they only rely on the data format. An example could be to answer "what's the area of my current state?" Comparison and Generation are about geometry relationships, which is the hardest part.

2.2 B. NoSQL (MongoDB)

MongoDB is a document oriented datastore. It is written in C++ and its main focus lies on high performance and retaining some friendly properties of SQL.

Currently, MongoDB uses GeoJSON objects to store spatial geometries. GeoJSON is an open-source specification for the JSON- formatting of shapes in a coordinate space. The GeoJSON spec is used in the geospatial community and there is growing library support in most popular languages.

Each GeoJSON document (or sub document) is generally composed of two fields:

1. Type – the shape being represented, which informs a GeoJSON reader how to interpret the “coordinates” field.

2. Coordinates – an array of points, the specific arrangement of which is determined by “type” field.

Overview of MongoDB's current implementation

- There are multiple geospatial indexes per collection in MongoDB- 2d, 2dsphere and geoHaystack
- Data can be imported from CSV files by converting it into GeoJSON objects.
- However MongoDB does not support R-trees.

Currently in MongoDB, containment and intersection tests are supported for both flat and spherical geometries. Sorting by distance is supported too. This is because these operations are the most popular operational queries for a modern application. They meet most requirements of typical location based applications.

3. COMPARATIVE ANALYSIS OF SPATIAL FUNCTIONS

Each of the implementation of spatial databases has a different approach for handling geometries.

3.1 Geometry / Flat Shape

PostGIS supports various operations for geometry , while MongoDB only supports storing points and containment / intersection queries with box/circle/polygon. 2d index is designed to speed up this case. Although it is possible to leverage open source libraries like GEOS, MongoDb don't plan to expand the features for flat geometry, because most LBS applications work with GPS data, which nicely fits with spherical model.

3.1 Geography / Spherical Shape

MongoDB supports storing and querying spherical shapes in GeoJSON. 2dsphere index is for this case. PostGIS only supports containment / intersection natively, which are supported by MongoDB as well. Distance between two objects is also supported in PostGIS.

4. DATASETS

For this analysis the dataset was custom generated.

4.1 Line Intersection

The Line Intersection problem works for any geometry in the space and report whether or not that geometry is being intersected by some other geometry. This problem is very important in real time scenarios for example - we want to know on which side of the road are we and also to avoid accidents we need intersection of various geometries. So we generated several synthetic datasets keeping in mind worst case scenarios.

The dataset consisted of two independent layers of horizontal lines and vertical lines with incremental lengths and their size varied from ten lines to ten million lines in each layer. Each line intersects at least one line in the other layer with some lines intersecting multiple lines. The above dataset emulates most of the real time cases with worst case scenarios. All the data in the analysis was processed using In-memory and no secondary memory was used.

4.2 Point Containment Problem

Like the previous problem Point Containment problem also works for any geometry and reports whether the given geometry is completely inside another geometry or not. This is a very important and popular problem with respect to spatial databases. It is useful in the domain of map generations, modeling, analyzing spatial data over a area for example we want to report how the number of houses have changed over a time in a city by analyzing the spatial data of that city. We can count the number of points(which represents each house in this case) in the polygon(city) for all the years using point within a polygon problem.

The dataset consisted of two independent layers, one of horizontal lines of incremental length and the size of the layer varied from 10 lines to 10 million lines. Another layer consisted of square boxes of different perimeter scattered both sequentially and randomly over the space with few lines completely inside the box, few intersecting the box and few completely outside the box. All the data in the analysis was processed using In-memory and no secondary memory was used.

5. PERFORMANCE

Tests were run on both of the database systems with same datasets one with Index and other without index. The time recorded are in seconds for both indexed and non-indexed analysis which are mentioned in table 1 below.

5.1 Line Intersection

Table 1. Time vs Intersection Table

No. of Intersections	MongoDB		PostGIS	
	Non-Index	*Index*	*Non-Index*	*Index*
21	1	1	1.546	9.721
1875	18	20	88.695	48.461
195691	185	190	13364.596	1963.123
< (1000*1000)	4093	3140	>1500000	172048

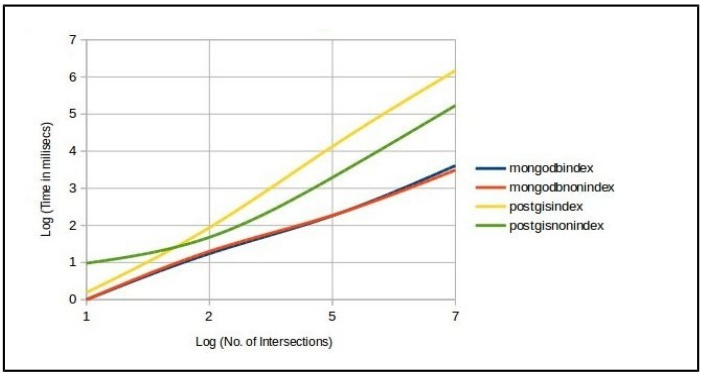

Figure 1. Graph of Intersections vs time (for all implementations)

5.2 Point Containment Problem

Table 1. Points within a polygon vs Time for all implementation.

No of Points Within	Mongo Non-Index	Mongo Index	PostGIS Non-Index	PostGIS Index
5002	6.75	7.75	52.95	54.97
5471	12	12	85.422	77.508
17004	20.8	24	142.53	158.76
45999	27.66	27.33	160.02	168.23
46008	50	51	675.33	631.556
122006	51.667	53.33	1339.23	1110.42
122016	96	99	2783.66	2609.47
204024	99.667	106	3991.564	3121.11223
298.23	125	112	-	-

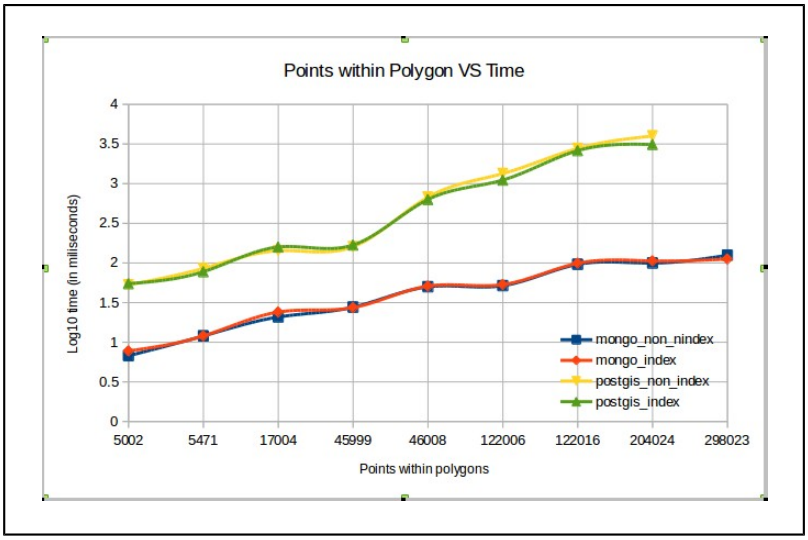

Figure 1. Graph of Point within Vs Time

6. RESULTS

Firstly the performance was analyzed without indexing any geometry and time was observed which suggests that MongoDB performs better as the datasize increases whereas PostGIS fails at very large datasets and time complexity increases exponentially. But after indexing the geometries performance of both the database engine improved by a substantial factor.

If we observe the results above we will notice that indexed datasets performs better than non indexed dataset and PostGIS time increases exponentially as size of dataset increases whereas MongoDB still performs within some bounds.

7. DISCUSSIONS AND CONCLUSIONS

The above analysis was necessary to evaluate the performance of NoSQL with respect to spatial databases and to tell whether NoSQL performs better than SQL in our current scenario. These results suggest that MongoDB performs better by an average factor of 25x which increases exponentially as the data size increases in both indexed and non-indexed operations for both LineIntersection and Point Containment problem. Given these results NoSQL databases may be better stated for simultaneous multiple-user query systems including Web-GIS and mobile-GIS.

This implies that non-relational databases are more suited to the multi-user query systems and has the potential to be implemented in servers with limited computational power. Further studies are required to identify its appropriateness and incorporate a range of spatial algorithms within non-relational databases. In future we are planning on expanding our study to other spatial query functions as well as spatial algorithms such as shortest problem to evaluate the performance of NoSQL on such platforms.

8. ACKNOWLEDGMENT

We would like to acknowledge the responses and interactions with MongoDB developers through emails which helped during this research project.

9. REFERENCES

References in the text should follow the standard format (Brown, 1997). As Brown (1997) suggested, they should be in the format shown immediately below. References are in 11 pt type.

Journal titles should be in full. Journal and book titles should be italicised. Multiple authors should all be listed.

Baas, B. Quak, W. Van Oosterom, P. De By, *NoSQL spatial: Neo4j versus PostGIS*

Cláudio de Souza Baptista, Maxwell Guimarães de Oliveira ,Tiago Eduardo da Silva. *Using OGC Services to Interoperate Spatial Data Stored in SQL and NoSQL Databases*

Dr. Johannes Scholz, TU Vienna, Gusshausstr. *Coping with Dynamic, Unstructured Data Sets – NoSQL: a buzzword or a savior?*

Zhifeng Xiao, Yimin Liu. *Remote sensing image database based on NOSQL database.*

http://nosql.mypopescu.com/post/300199706/geo-nosql-couchdb-mongodb-tokyo-cabinet

AT2-01

Geographical Weighted Regression Model for Improved Near-shore Water Depth Estimation from Multispectral Imagery

Poliyapram Vinayaraj[1] **, Venkatesh Raghavan**[1] **and Shinji Masumoto**[2]
[1]Media Center, Osaka City University
3-3-138 Sugimoto, Sumiyoshi-ku, Osaka 558-8585, Japan
[2] Graduate School of Science, Osaka City University, Japan
Email: vinay223333@gmail.com

ABSTRACT

There is often a need for making a high-resolution or a complete bathymetric map based on sparse point measurements of water depth. The common practice of previous studies has been to calibrate a single global depth regression model for an entire image. The performance of conventional global models is limited when the bottom type and water quality vary spatially within the scene. For a more accurate and robust water-depth mapping, this study proposes a regression model for a geographical region or local area rather than using a global regression model. Here , two different methods were discussed according to the availability of reference depth data to ensure the accuracy of the Geographical Weighted Regression (GWR) model. The entire data analysis workflow was carried out using GRASS GIS Version 7.0.1. GWR is seen to be effective in addressing the problem introduced by heterogeneity of the bottom type and provide better bathymetric estimates in near coastal waters. The study was carried out at Puerto Rico, Northeastern Caribbean Sea. High spatial resolution (RapidEye) and Medium spatial resolution (Landsat 8) images were used to estimate the depth. in order to calibrate and evaluate estimated depth, collected high accuracy LiDAR depth data (4m resolution) provided by NOAA. Comparison of results indicates that the GWR model improves the depth estimation significantly. The maximum depth calculated by this methid is 20 meter. In order to estimate the accuracy of the results in terms of quantitative analysis have been carried out by R,R^2 and RMSE. The results reveal the significant improvements by GWR model than Global model from both Landsat 8 [GWR model ($R=0.97,R^2=0.93$,RMSE=1.41m), Global model ($R=0.88,R^2=0.78$,RMSE=2.63m)]and RapidEye [GWR model ($R=0.97,R^2=0.93$,RMSE=1.34m), Global model ($R=0.88,R^2=0.78$,RMSE=2.60)] data.

1. INTRODUCTION

Accurate and high-resolution bathymetric data are a necessity for a wide range of coastal oceanographic research, but become particularly critical in predictive modeling approaches for the proactive management of impacts on near-shore. Mapping shallow water depth data from ships by Sonar is a quite an expensive task. Many shallow water areas are not accessible by hydrographic ships due to rocks, coral reefs or simply due to inaccessibility by boats due to shallow depth of the water. Airborne Light Detection and Ranging (LiDAR) can provide complete and accurate bathymetric measurements in shallow areas, but availability of this technology is currently limited and also involves significant cost. Thus, alternative method to estimate high resolution depth data need to be devised and validate. Until now, several algorithms have been proposed to estimate bathymetry from single band [5] and multispectral bands (Lyzenga, 2006, Stoffle, 1991 and Stumpf, 2004). The multispectral algorithms were commonly practiced to estimate good accuracy depth estimates by addressing the problem introduced by heterogeneous bottom type. Therefore this study was carried out to investigate depth estimation from multispectral imagery by conventional global regression and Geographically Weighted Regression (GWR) model.

2. MATERIALS AND METHODOLOGY

The study was carried out at Puerto Rico, northeastern Caribbean Sea. Two different satellite data were collected in order to test the algorithm with high and moderate spatial

resolution data (RapidEye and Landsat 8). RapidEye data has 12-bit radiometric resolution and 5 meter spatial resolution. Even though the Landsat 8 data also has 12-bit radiometric resolution, it provides 30m spatial resolution. In order to calibrate and evaluate estimated depth, collected high accuracy LiDAR depth data (4m resolution) provided by NOAA (https://catalog.data.gov/dataset/). Five spectral bands collected from Landsat data are used for depth estimation (visible band (0.43-0.67μm)and Near-infra red band (0.85-0.88μm)) and Short wave infrared band is effectively used for correction as shown in equation 1. In the case of RapidEye data four visible bands (0.44-0.73μm) were used for estimation and Near infrared band (0.76-0.85μm) used for correction as shown in equation 1.

2.1 Pre-proccesing

The radiance observed from space using a visible light sensor over optically shallow water primarily consists of four components: atmospheric scattering component (path radiance), surface reflection component, in-water volume scattering component, and bottom reflection component. Primarily, a correction method is applied remove atmospheric scattering component, surface reflection component and in-water volume scattering component to retrieve only the bottom reflection component which is linearly related to depth Kanno, 2012). The correction algorithm (shown below) is a modification to the Kanno *et al.*, 2012.

$$X(\lambda)i = log\ [(L\ \lambda\ i - \alpha 0 - \alpha 1\ L\lambda IR)/L\lambda\ i] \quad (1)$$

Where, X(λ) i is the log transformed corrected band i, L λi is the radiance of band i, α0, α1 are the coefficients, LλIR is the radiance of infrared band. More detailed description about the algorithm is given in (Vinayaraj *et al.*, 2014). Here, LλIR was the infrared band used to correct the atmosphere, water surface and water column reflectance. The wavelengths used for correction of Landsat 8 and RapidEye images are 1.57-1.65 μm and 0.76 – 0.85 μm respectively. The previous study (Vinayaraj *et al.*, 2014) reveals that short wave infrared band is better to correct the radiance than near infrared band. Since the RapidEye image does not have a short-wave infrared band Near infrared band (0.76–0.85 μm) has been used for correction process.

2.2 Depth estimation

The two different method applied in this study are the conventional global regression model and GWR model.

2.2.1 *Global regression model*

The conventional regression models are a common practice by several authors (Kanno, 2012 and Lyaenga, 2006) in order to address the heterogeneity by bottom type and water quality. It is a simple multiple linear regression where, multispectral bands were taken as independent variables. The equation is shown below,

$$h=\beta 0+\beta 1X(\lambda)1+\beta 2X(\lambda)2+\ldots\ldots+\beta nX(\lambda)m \quad (2)$$

Where, h is the estimated depth, β0 and βn are the intercept and slope derived from the multiple regression. X(λ)n is the log transformed bands (derivation is shown in equation (1)). The conventional global regression model uses a single set of coefficients (β0, βn) for the whole image.

2.2.1 *GWR model*

To address the inadequacy of the global bathymetric inversion model, we apply the GWR model that was originally was proposed by (Brunsdon, 1996) for improved depth

retrieval in complex and heterogeneous near coastal environments. The spatial non-stationarity in the image scene was addressed by this method. Similar to the kernel regression method, the GWR model uses a window to define the local neighborhood. Kernel size was decided according to the density of the reference depth and pixel size of the satellite imagery used. The equation is denoted as follows,

$$hj=\beta 0j+\beta 1jX(\lambda)1j+\beta 2jX(\lambda)2j+\ldots\ldots+\beta njX(\lambda)mj \quad (3)$$

Where, j is the respective pixel, therefore coefficients for each pixel (j) was derived by assigning a fixed bandwidth. The bandwidth represents the size of the kernel used for regression. The pixels closer to the kernel center were assigned higher weights. The script called "r.gwr" available in GRASS GIS Version 7.0.0. was used to calculate GWR model.

3. DATA ANALYSIS

Data analysis in this research is carried out by considering the reference depth at different level in such a way that the different cases of common availability of reference depth could be addressed. Meaning that commonly available reference depth data are of two types, one is high-resolution echo sounder or airborne LiDARs available for a small region and the second one is low resolution data extracted from bathymtric charts. The reference depth used in this study for depth estimation analysis was derived from LiDAR survey. Study consider these scenarios of reference depth availability and evaluate the performance of both GWR model and Global model in order to estimate the near-shore depth from Landsat 8 and RapidEye images. Scenario 1 is to estimate depth from randomly distributed reference depth points, scenario 2 is to estimate the depth from 300 meter interval reference depth points, scenario 3 is to estimate depth from 600 meter interval reference depth points and scenario 4 is to estimate (extrapolate) depth from reference depth point available at small area to a complete near-shore (<=20m depth) region covering by the image scene. This scenario 4 is basically a new method proposed by this study to estimate the bottom class based coefficients and derive depth where no reference depth available. By using maximum likelihood analysis bottom classification of the study area is carried out and six classes were derived. Then, coefficients for each class is determined by averaging the coefficients derived from GWR model from each classes. Further, these class based coefficients are used extrapolate the depth from small region to all study area where no reference depth available. Here onwards, scenario numbers mentioned above are used to describe results in each case.

4. RESULT AND DISCUSSION

Previous studies have been observed (Vinayaraj *et al.*, 2014) that the global model is not efficient to address the non-stationarity in the near-shore due to the difference bottom type and water quality. Also observed that global model using single coefficients derived from each band cannot be sufficient to address the difference in the reflectance of each band. Hence, this study utilize the efficacy of GWR model in addressing the non-stationarity (Brunsdon, 1996) to near-shore water depth estimation algorithm. Here, four different scenarios were tested to evaluate performance of the GWR model and compare the derived results with different scenarios were carried out. The minimum and maximum of the reference depth value used for calibration of algorithms was set to 1m and 20m respectively. The minimum set as 1m in order to avoid error would have occurred due to tidal variation between data sets processed. The evaluation tests of the estimated results are carried out by using randomly selected data which were not included in the calibration or estimation process.

Scenario 1: Randomly selected 60,000 and 10,000 reference depth points were selected

and used for calibration and coefficient determination from RapidEye and Landsat 8 data respectively. Scenario 1 was carried out to examine the efficacy of GWR model in estimating depth in near-shore waters.

Table 1. Results of evaluation of depth estimates derived from Landsat 8 and RapidEye data at Scenario 1

Satellite data	Global model			GWR model		
	R	R^2	RMSE (m)	R	R^2	RMSE (m)
Landsat 8	0.88	0.78	2.63	0.97	0.93	1.41
RapidEye	0.88	0.78	2.48	0.97	0.93	1.34

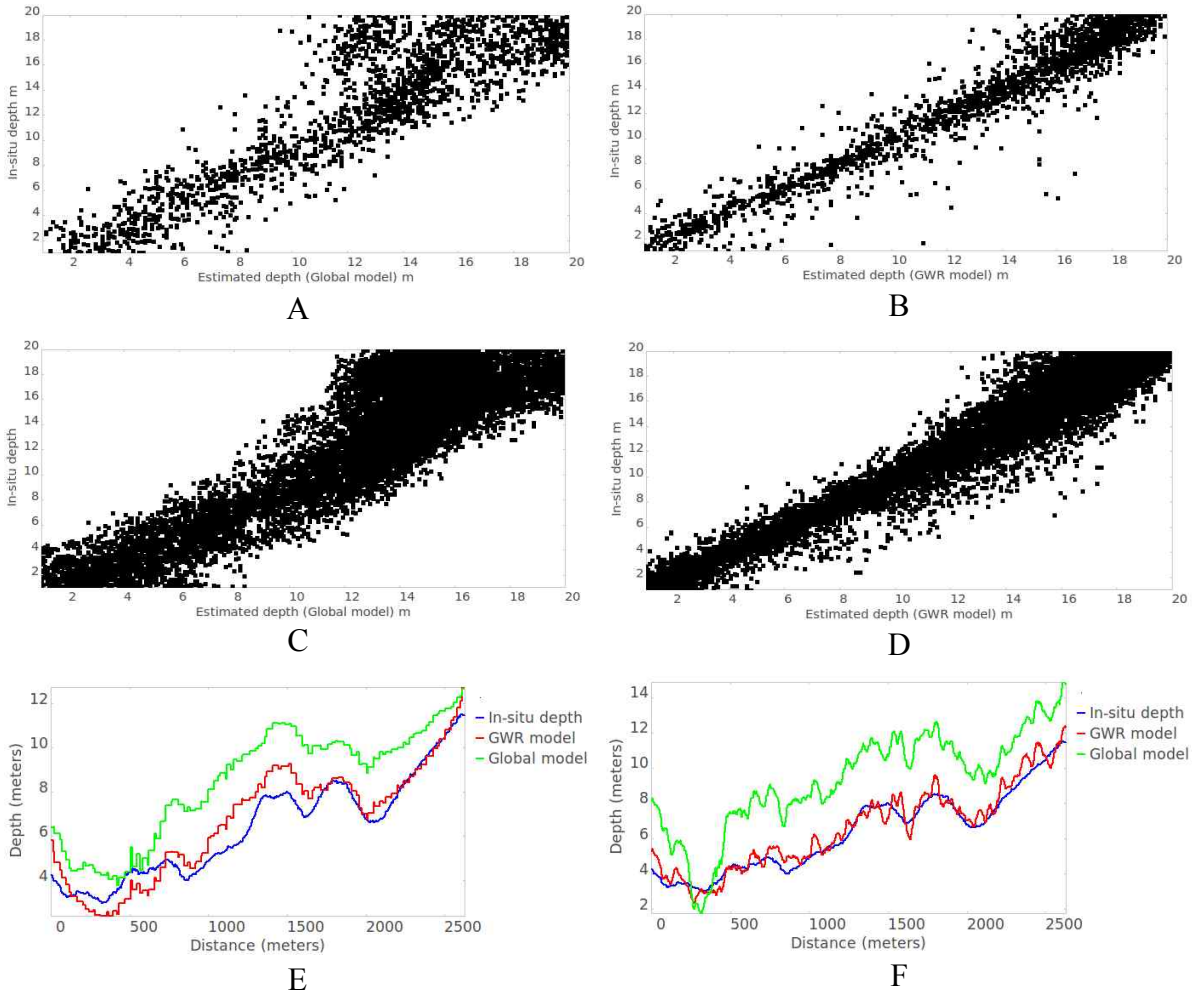

Figure 1. Evaluation of estimated depth derived from Landsat 8 and RapidEye data with reference depth for Scenario 1. Scatter plot of Landsat 8 (Global (A), GWR(B)), scatter plot of RapidEye (Global (C), GWR(D)) and profile of drawn on Landsat 8 (E) and RapidEye (F).

The results shown in Table 1 derived from both Landsat and RapidEye demonstrate the efficacy of GWR model. The comparative study has shown that GWR model improves the accuracy of the estimation significantly than the conventional global model. A bi-variate scatter plot shown in Figure1 and demonstrate the strong linear relationship with reference depth in GWR than and global model. Profile of the depth estimate also suggest that GWR

profile is closer to the reference depth than global model.

Scenario 2: reference depth points were collected at an interval of 300 meter between the neighboring pixel for both Landsat 8 and RapidEye. 1750 were used to estimate the depth by both GWR model and global model and evaluated. The result of evaluation tests shown in Table 2 suggests that the GWR model significantly improve the accuracy even though less number of reference depth available. In Figure 2, bi-variate scatter plot of GWR is showing strong linear relationship with reference depth than global model. Figure 2 shows the profile of the depth estimates, demonstrate that GWR model more synchronize with reference depth than global model.

Table 2. Results of evaluation of depth estimates derived from Landsat 8 and RapidEye data at Scenario 2

Satellite data	Global model			GWR model		
	R	R^2	RMSE (m)	R	R^2	RMSE (m)
Landsat 8	0.88	0.78	2.63	0.96	0.91	1.72
RapidEye	0.89	0.78	2.51	0.95	0.91	1.61

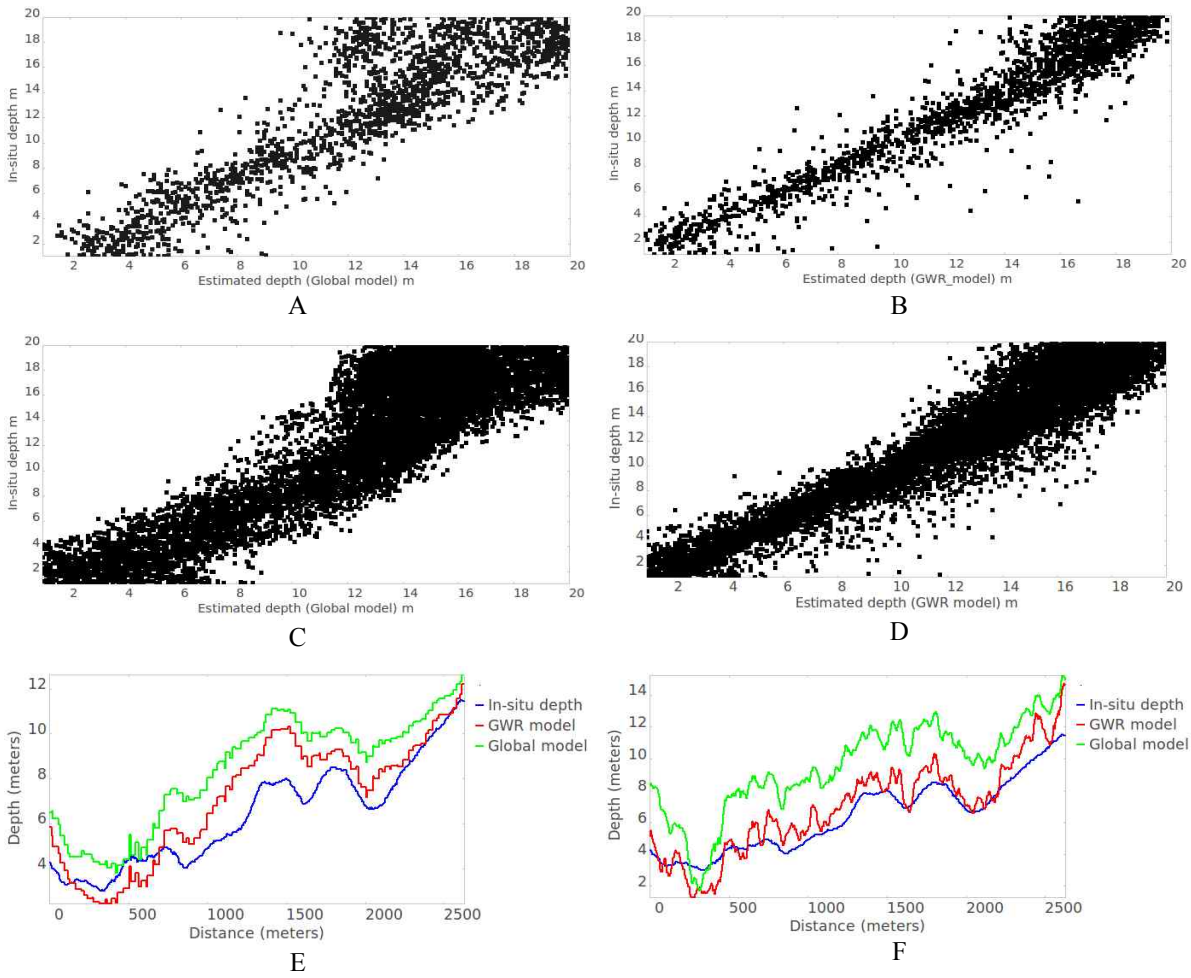

Figure 2. Evaluation of estimated depth derived from Landsat 8 and RapidEye data with reference depth for Scenario 2. Scatter plot of Landsat 8 (Global (A), GWR(B)),

scatter plot of RapidEye (Global (C), GWR(D)) and profile of drawn on Landsat 8 (E) and RapidEye (F).

Scenario 3: reference depth points were collected at an interval of 600 meter between the neighboring pixel for both Landsat 8 and RapidEye. Only 450 reference depth points were sufficient to cover all the study area. Even though the counts of reference depth points very less and interval between the neighboring pixels are far, the evaluation of the results showing that the GWR model is providing better estimates (Table 3). Bi-variate scatter plot in Figure 3 also show that GWR model estimates has strong linear relationship with reference depth. Observation carried out by a profile in Figure 3 (E, F) also indicate the improvement in accuracy of GWR model.

Table 3. Results of evaluation of depth estimates derived from Landsat 8 and RapidEye data at Scenario 3

Satellite data	Global model			GWR model		
	R	R^2	RMSE (m)	R	R^2	RMSE (m)
Landsat 8	0.88	0.78	2.62	0.94	0.88	1.91
RapidEye	0.88	0.78	2.48	0.93	0.87	1.87

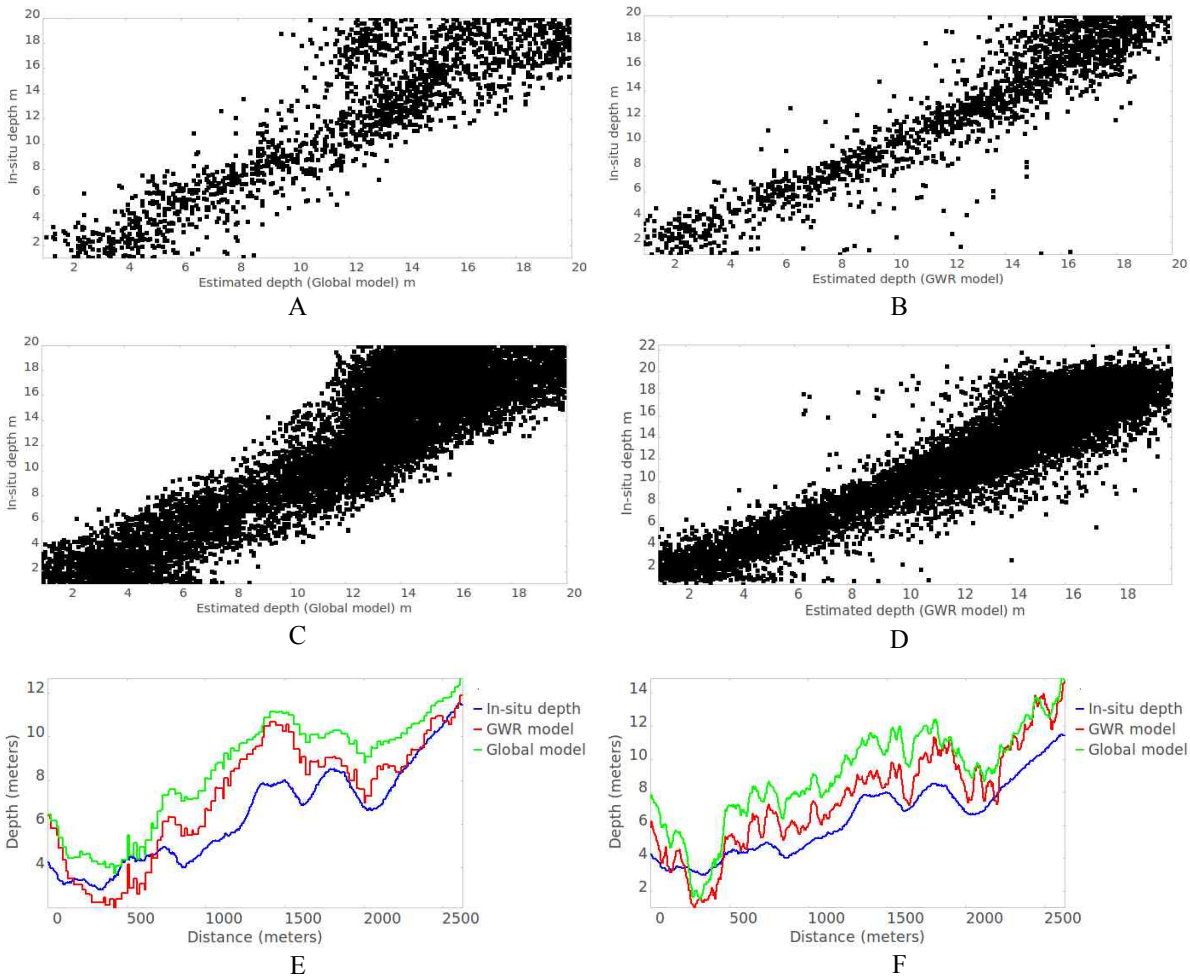

Figure 3. Evaluation of estimated depth derived from Landsat 8 and RapidEye data with reference depth for Scenario 3. Scatter plot of Landsat 8 (Global (A), GWR(B)),

scatter plot of RapidEye (Global (C), GWR(D)) and profile of drawn on Landsat 8 (E) and RapidEye (F).

Table 4. Results of evaluation of depth estimates derived from Landsat 8 and RapidEye data at Scenario 4

Satellite data	Global model			GWR model		
	R	R^2	RMSE (m)	R	R^2	RMSE (m)
Landsat 8	0.83	0.68	3.49	0.87	0.75	2.86
RapidEye	0.88	0.78	2.60	0.90	0.81	2.26

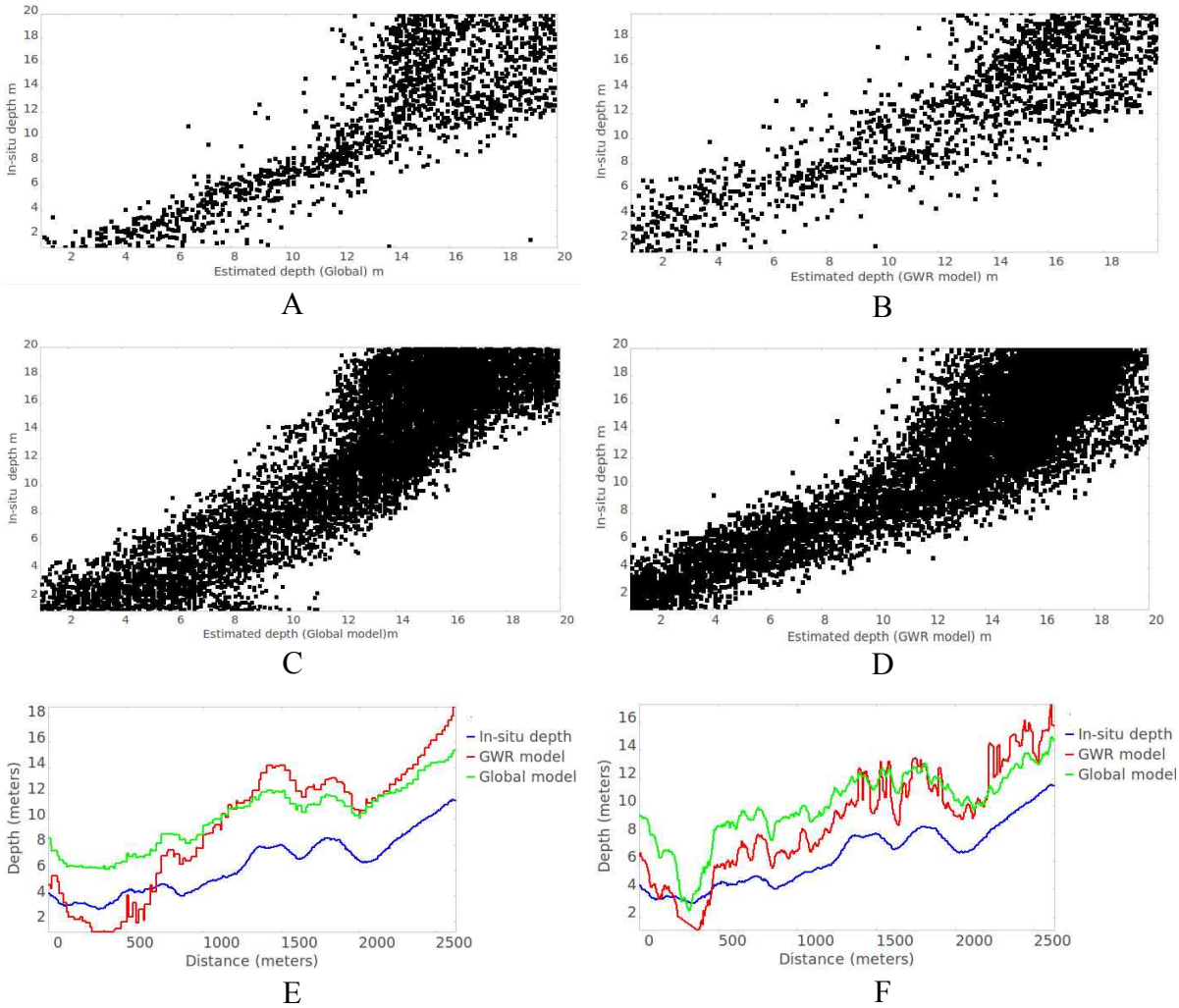

Figure 4. Evaluation of estimated depth derived from Landsat 8 and RapidEye data with reference depth for Scenario 4. Scatter plot of Landsat 8 (Global (A), GWR(B)), scatter plot of RapidEye (Global (C), GWR(D)) and profile of drawn on Landsat 8 (E) and RapidEye (F).

Scenario 4: The randomly distributed 20000 reference depth points for RapidEye and 3300 for Landsat 8 were collected from small area to extrapolate/estimate depth for entire study area even where no reference depth depth available. The method proposed by this study

is efficient to extrapolate depth from bottom class based coefficients. The comparative quantitative analysis shown in the Table 4 indicate that results are better than global model in terms of R, R^2 and RMSE. Figure 4 (B, D) is showing strong linear relationship with reference depth than global model (A, C). Even though the quantitative analysis and bivariate scatter plot shows improvement in accuracy of GWR model, profile of the GWR model is not showing such improvements than global model.

Many previous studies have demonstrated that optical remote sensing imagery can be employed to derive water depth estimates. The widely used approach is to calibrate one global inversion model for the entire image scene and then apply the same inversion model to every image pixel for bottom depth retrieval. The intention of the study was to explore the efficacy of GWR model in addressing spatial non-stationarity in the near-shore water region in order to estimate water depth. The evaluation of the results derived by four different scenarios shows significant improvements in both quantitative and qualitative terms. The results provided by GWR model in Scenario 1, Scenario 2 and Scenario 3 shows the significant improvement over global model. In case of Scenario 4, a method originally proposed in this study is also providing better accuracy results in terms of quantitative analysis than global model. But the comparative study carried out by profile (Figure 4 (E, F) is showing that the results has to be improved further. Therefore the further research will be focused to estimate optimal number of bottom classes for improved depth estimation.

8. REFERENCES

Brunsdon, A. C., Fotheringham, S., and Charlton, E. M., 1996. *Geographically Weighted Regression: A Method for Exploring Spatial Nonstationarity*, Geographical Analysis, 28 (2), pp. 283-297.

Lyzenga, D. R., Malinas, N. R., and Tanis,F. J., 2006. *Multispectral bathymetry using a simple physically based algorithm*, IEEE Transaction on Geoscience and Remote Sensing., 44(8), 2251–2259.

Stoffle R. W., and Halmo,D. B., 1991.*Satellite monitoring of coastal marine ecosystems: A case from the Dominican Republic*, Consortium for Integrated Earth Science Information Network (CIESIN). Sagin aw, Michigan.

Stumpf, R. P., Holderied, K., and Sinclair, M., 2004. *Determination of water depth with high-resolution satellite imagery over variable bottom types*. Limnology and Oceanography., 48 (2), 547–556

Vinayaraj, P., Raghavan,V., Masumoto, S., Johnson, G., and Manessa, M. D. M., 2014. I*nvestigation of algorithm to estimate shallow water bathymetry from Landsat-8 images*, International symposium on Geinformatics for Spatial-Infrastructure in Earth and Allied Sciences GIS-IDEAS, Vietnam, 465-470,