HOTINE OBLIQUE MERCATOR MAP PROJECTION IN PROJ4JS LIBRARY

K. Anshar , N. Suryana , Othman, M.F.I.

Centre for Advanced Computing Technology, Faculty of Information and Communication Technology,
Universiti Teknikal Malaysia Melaka, Melaka, Malaysia
p031420004@student.utem.edu.my, (nsuryana, mohdfairuz)@utem.edu.my

KEY WORDS: GIS Application, Hybrid Application, Coordinate Transformation, Hotine Oblique Mercator, Proj4JS

ABSTRACT:

The implementation of GIS application using hybrid approach on smartphone should have a coordinate transformation capability from one coordinate system into another. One of the available libraries to handle a coordinate transformation is Proj4JS. Unfortunately, this library is not able to perform coordinate transformation from Hotine Oblique Mercator into other projection. One example of this projection is Kertau (RSO) / RSO Malaya (m). We proposed new approach that introduce new formula, algorithm, Proj4JS definition and involve datum transformation to enable the coordinate transformation from Hotine Oblique Mercator into another coordinate system or vice versa. Using the proposed approach, all the features are rendered on top the feature that rendered using Spherical Mercator projection. We compare the result also with the Google Map. It shows that all features are properly rendered on top of Google Map features.

1. INTRODUCTION

1.1 Background

Developing a hybrid mobile application using two different frameworks has been discussed in (Anshar K. and Suryana N., 2014). Each framework provides a different approach that allows the web technology to communicate with the native technology. Moreover, (Anshar K. et al. 2015) shows that the hybrid application framework File API allows the application to read Geographic Information Systems (GIS) data stored as a text file in the Smartphone local system, such as GML, KML, GeoRSS, and GeoJSON.

With the proliferation of current Web Maps technology, there are many GIS data processes which are performed in the server including the transformation from one coordinate system into another. In order to support access to different GIS data source including from the Smartphone Local System, the GIS Application for Smartphone should have the capability to perform coordinate transformation from one coordinate system into another. One of the available libraries used by the browser engine to handle coordinate transformation is Proj4JS. Unfortunately, this API is unable to perform coordinate transformation from Hotine Oblique Mercator into other projection (Proj4JS User Guide, OSGeo Foundation, 2012). One example of this projection is **Kertau (RSO) / RSO Malaya (m)**.

In order to overcome this problem, an algorithm and formula to handle the coordinate transformation process are proposed and applied in Proj4JS API. The coordinate transformation result using the proposed approach is compared with the result using other coordinate system. These two maps are displayed together using different layers in order to perform the data analysis process. All the layers are transformed into the same coordinate system i.e. "**EPSG:900913**". To provide a comprehensive data analysis, tables are tabulated to present the deviation between proposed approach and Geotools API.

1.2 OpenLayers and Proj4JS Library

GeoJSON and GML store the map projection information on their file content. As described in the OpenLayers API documentation, "When using vector layers with strategies, layer projection should be set to the projection of the data source if that is different from the map default. If it is not specified in the layer options, it is set to the default projection specified in the map, when the layer is added to the map" (OpenLayers.Layer constructor, OpenLayers, 2012).

The OpenLayers library has the ability to transform coordinates between Geographic ("EPSG:4326") and Spherical Mercator ("EPSG:900913") projection only. To handle other map projections, it requires Proj4JS library. Currently this library is unable to handle Hotine Oblique Mercator projection (Proj4JS User Guide, OSGeo Foundation, 2012). This paper proposes a new approach to enable coordinate transformation from Hotine Oblique Mercator into geographic ("EPSG:4326") or Spherical Mercator ("EPSG:900913") projection which is applied into Proj4JS library.

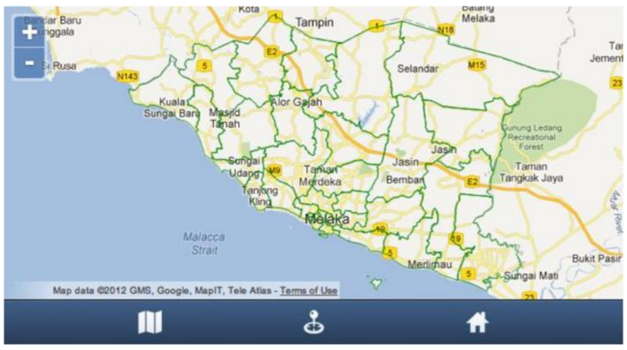

Figure 1. The Projection of Data Source is Spherical Mercator

To get a clear view of this problem, the discussion starts by displaying a map that has overlay layers. These overlay layers consist of same geographic features with different projections. The projection of OpenLayers map is set to the Spherical Mercator projection. In Figure 1, the projection of the data source is Spherical Mercator. In Figure 2, the projection of the data source is Hotine Oblique Mercator. In Figure 2, the geographic features are rendered on different coordinates with Figure 1. Figure 2 shows that Proj4JS is not able to transform a coordinate from Hotine Oblique Mercator into Spherical Mercator.

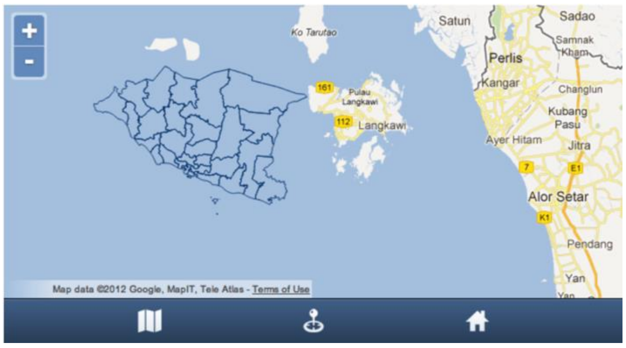

Figure 2. The Projection of Data Source is Hotine Oblique Mercator

2. HOTINE OBLIQUE MERCATOR

This discussion starts on the Hotine Oblique Mercator projection based on “Coordinate Conversions and Transformations including Formulas” document released on April 2012 by The International Association of OGP Geomatics Committee (OGP Publication 373-7-2, OGP, 2012). This map projection is shown in Figure 3.

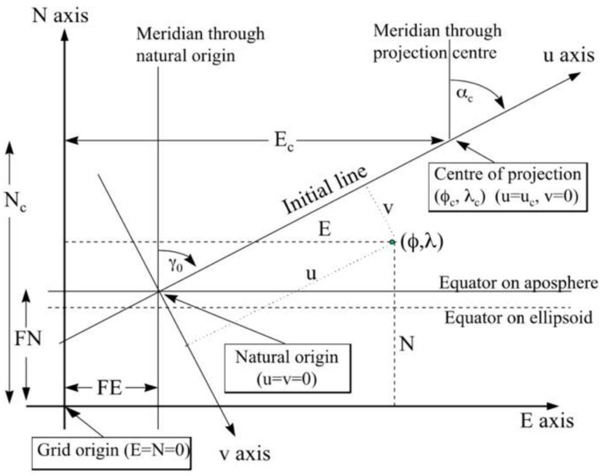

Figure 3. Hotine Oblique Mercator projection

(Source: OGP Publication 373-7-2)

To determine the map projection method with referring to Figure 3, this document describes that, “if the false grid coordinates are defined at the intersection of the initial line and the aposphere (the equator on one of the intermediate surfaces inherent in the method), that is at the natural origin of the coordinate system, the map projection method is known as variant A”. ” If the false grid coordinates are defined at the projection centre the projection method is known as variant B”. Table 1 describes the map projection method based on its parameter. This table can be used as a reference to build a simple algorithm to determine the Map Projection Method as shown in Figure 4.

	Method		
	Hotine	**Hotine**	**Laborde**
	Variant A	**Variant B**	
EPSG Dataset coordinate operation method code:	**9812**	**9815**	**9813**
Parameter			
Latitude of the projection center (φ_C)	v	v	v
Longitude of the projection center (λ_C)	v	v	v
Azimuth of the initial line (α_C)	v	v	v
Angle from the rectified grid to the skew (oblique) grid (γ_C)	v	v	
Scale factor on the initial line (k_C)	v	v	v
False Easting (easting at the natural origin) (FE)	v		
False Northing (northing at the natural origin) (FN)	v		
Easting at the projection center (E_C)		v	v
Northing at the projection center (N_C)		v	v

Table 1. Determining the Map Projection Method Based on the Parameter

(Source: OGP Publication 373-7-2)

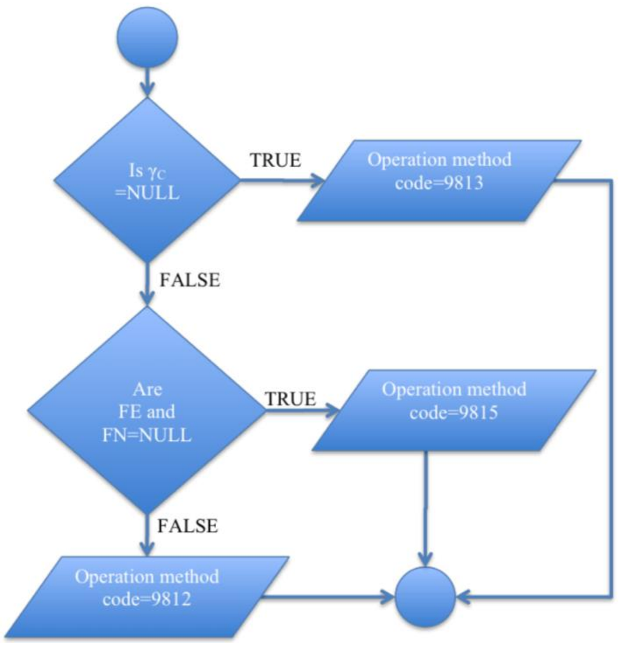
Is γ_C =NULL
TRUE
Operation method code=9813
FALSE
Are FE and FN=NULL
TRUE
Operation method code=9815
FALSE
Operation method code=9812

Figure 4. Algorithm to Determine the Map Projection Method Based on The Parameter

Proj4JS library version 1.1.0 provides *Proj4JS.Proj* constructor to define the map projection parameter and its value. This constructor has two arguments as input parameter. The first argument value is used to determine the map projection method. The argument value can be in Open Geospatial Consortium (OGC) Well Known Text (WKT) or Map Projection Code.

If the value is Map Projection Code then the *Proj4JS.defs* is used to get the Proj4JS format definition value as can be seen in Listing 1. *Proj4JS.Proj* is able to extract the argument value to get all the parameters and its value from the OGC WKT or Proj4JS format definition. The Human-Readable OGC WKT for “EPSG:3168” EPSG code is shown in Listing 2.

```
PROJCS["Kertau (RSO) / RSO Malaya (m)", GEOGCS
["Kertau (RSO)", DATUM [ "Kertau_RSO", SPHEROID
["Everest 1830 (RSO 1969)", 6377295.664, 300.8017,
AUTHORITY["EPSG","7056"]],
AUTHORITY["EPSG","6751"]], PRIMEM [ "Greenwich",
0, AUTHORITY["EPSG","8901"]], UNIT["degree",
0.01745329251994328, AUTHORITY["EPSG","9122"]],
AUTHORITY["EPSG","4751"]], UNIT["meter", 1,
AUTHORITY["EPSG","9001"]],
PROJECTION["Hotine_Oblique_Mercator"],
PARAMETER["latitude_of_center",4],
PARAMETER["longitude_of_center",102.25],
PARAMETER["azimuth",323.0257905],
PARAMETER["rectified_grid_angle",323.1301023611111],
PARAMETER["scale_factor",0.99984],
PARAMETER["false_easting",804670.24],
PARAMETER["false_northing",0],
AUTHORITY["EPSG","3168"],
AXIS["Easting",EAST], AXIS["Northing",NORTH]]
```

Listing 1. OGC WKT for “EPSG:3168”

```
Proj4JS.defs["EPSG:3168"] = "+proj=omerc +lat_0=4
+lonc=102.25 +alpha=323.0257905 +k=0.99984
+x_0=804670.24 +y_0=0 +a=6377295.664
+b=6356094.667915204 +units=m +no_defs";
```

Listing 2. The Proj4JS format definition of “EPSG:3168”

Both of these parameters are evaluated using the algorithm as shown in Figure 4 to determine the map projection method code. The transformation process for “EPSG:3168” should use “Hotine Variant A” or “9812” operation method code. The result for OGC WKT is “9812”, which is correct. The result for this Proj4JS format definition is “9813”, which is not correct. This is due to the Proj4JS format definition which does not have “rectified_grid_angle” parameter and its value. To fix it, the new parameter for “rectified_grid_angle” is added into the Proj4JS format definition as can be seen on Listing 3, which is highlighted in blue.

```
Proj4JS.defs["EPSG:3168"] = "+proj=omerc +lat_0=4
+lonc=102.25 +alpha=323.0257905
+lamda_c=323.1301023611111 +k=0.99984
+x_0=804670.24 +y_0=0 +a=6377295.664
+b=6356094.667915204 +units=m +no_defs";
```

Listing 3. The Proj4JS Format Definition of “EPSG:3168” with *lamda_c* Property

After applying:

- The formulas described in OGP Publication 373-7-2 as shown in Figure 3;
- Proposed algorithm as shown in Figure 4.
- The new Proj4JS format definition of “EPSG:3168” as shown in Listing 3.

into the Proj4JS library, we get the results as shown in Figure 5 and Figure 6. Both figures have different zoom levels. Using the higher zoom level, it shows that the geographic features on the “EPSG:3168” layer projection are still rendered on different coordinates with the “EPSG: 900913” layer projection. This problem is discussed in detail in the following section.

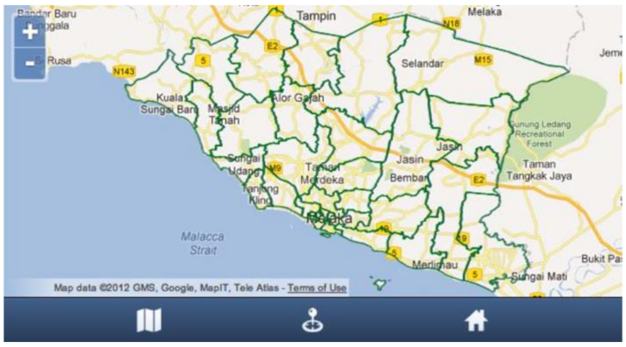
Figure 5. The Dun Melaka Using new Algorithm and Formula

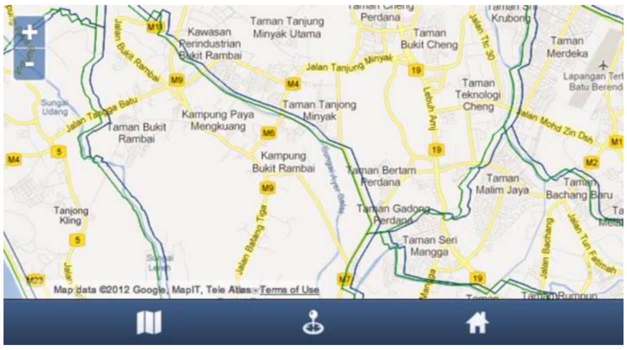
Figure 6. The Dun Melaka Using Higher Zoom Level

3. DATUM TRANSFORMATION

Two map projections may use different reference ellipsoids to produce a map for different areas on the earth surface. Therefore, these two map projections will have different latitude and longitude (R. Knippers, 2009). Each local reference ellipsoid has a datum shift (position compared) to global reference ellipsoid. Because these maps use different reference ellipsoid, they use different geodetic datum. Therefore, the transformation process should involve a datum transformation.

The discussion on datum transformation in Proj4JS starts with the algorithm of coordinate transformation applied on this library. Figure 7 shows the algorithm applied on Proj4JS to transform a point from one coordinate system to another. From this figure, some conditions should be fulfilled in order to perform the datum transformation.

This algorithm has three arguments as follows:

1. Source Map Projection.
   Target Map Projection.
2. A point to transform, may be in geodetic (long, lat) or Cartesian (x,y), but should always have 2 properties.

In the previous example, the projection of data source is Hotine Oblique Mercator. This layer is rendered on the map with a projection is set to Spherical Mercator. This means, the source projection is “EPSG:3168” and the destination projection is “EPSG:900913”. All the coordinates of geographic feature on the data source are transformed from “EPSG:3168” into “EPSG:900913”. Using the Map projection properties as shown in Appendix A, the coordinate transformation flow follows the blue arrow in Figure 7. On this figure the datum transformation is highlighted in green. A detailed process of datum transformation algorithm is shown in Figure 8.

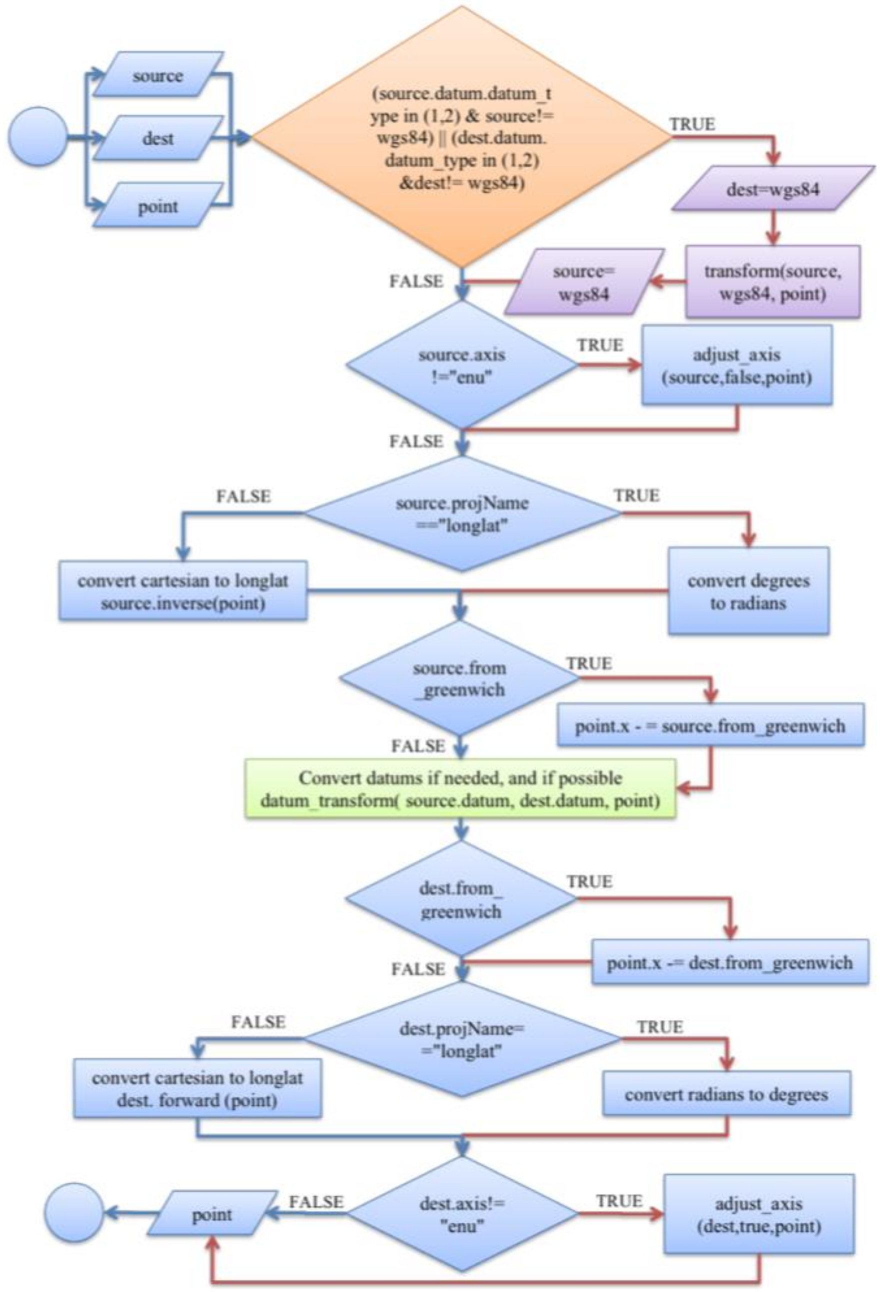
source
dest
point
(source.datum.datum_type in (1,2) & source!= wgs84) || (dest.datum.datum_type in (1,2) &dest!= wgs84)
TRUE
dest=wgs84
transform(source, wgs84, point)
source= wgs84
FALSE
source.axis !="enu"
TRUE
adjust_axis (source,false,point)
FALSE
FALSE
source.projName =="longlat"
TRUE
convert cartesian to longlat source.inverse(point)
convert degrees to radians
source.from_greenwich
TRUE
point.x - = source.from_greenwich
FALSE
Convert datums if needed, and if possible datum_transform(source.datum, dest.datum, point)
dest.from_greenwich
TRUE
point.x -= dest.from_greenwich
FALSE
FALSE
dest.projName= ="longlat"
TRUE
convert cartesian to longlat dest. forward (point)
convert radians to degrees
point
FALSE
dest.axis!= "enu"
TRUE
adjust_axis (dest,true,point)

Figure 7. Coordinate Transformation Algorithm in Proj4JS

Figure 8 shows that there are four processes on datum transformation i.e. Geodetic to Geocentric, Geocentric to WGS84, Geocentric from WGS84, and Geocentric to Geodetic. Based on the Map projection properties as shown in Appendix A and the algorithm as shown in Figure 8, it is evident that the coordinate transformation from “EPSG:3168” into “EPSG:900913” map projection does not involve the datum transformation.

Based on the algorithm shown in Listing 4, the datum transformation depends on *datum.datum_type* map projection property. This property depends on *datum.datum_params* property and this property depends on towgs84 property as can be seen in Figure 9. The default value of *datum.datum_type* property is *Proj4JS.common.PJD_WGS84* or 4. If the *towgs84* property does not have value then the datum.datum_type remains the same with the default value. The “EPSG:3168” Proj4JS format definition in Listing 3 does not have *towgs84* property; therefore, the *datum.datum_type* property value remains the same with the default value. To involve the datum transformation on the coordinate transformation then *datum.datum_type* value of the source or destination map projection property should be either 1 or 2. To get this value then the map projection should have *towgs84* property and a list of datum transformation parameters from local geodetic to World Geodetic System (WGS84) is shown in Appendix B.

```
initialize: function(proj) {
this.datum_type = Proj4JS.common.PJD_WGS84;
if (proj.datumCode && proj.datumCode == 'none') {
this.datum_type = Proj4JS.common.PJD_NODATUM;
}
[...]
if (proj.datum_params[0]!=0 || proj.datum_params[1]!=0 ||
proj.datum_params[2]!=0 ) {
this.datum_type = Proj4JS.common.PJD_3PARAM;
```

```
}
if (proj.datum_params.length > 3) {
if (proj.datum_params[3]!=0 || proj.datum_params[4]!=0 ||
proj.datum_params[5]!=0 || proj.datum_params[6]!=0 ) {
this.datum_type = Proj4JS.common.PJD_7PARAM;
[...]
```

Listing 4. Inilitilizing Projection Datum Class

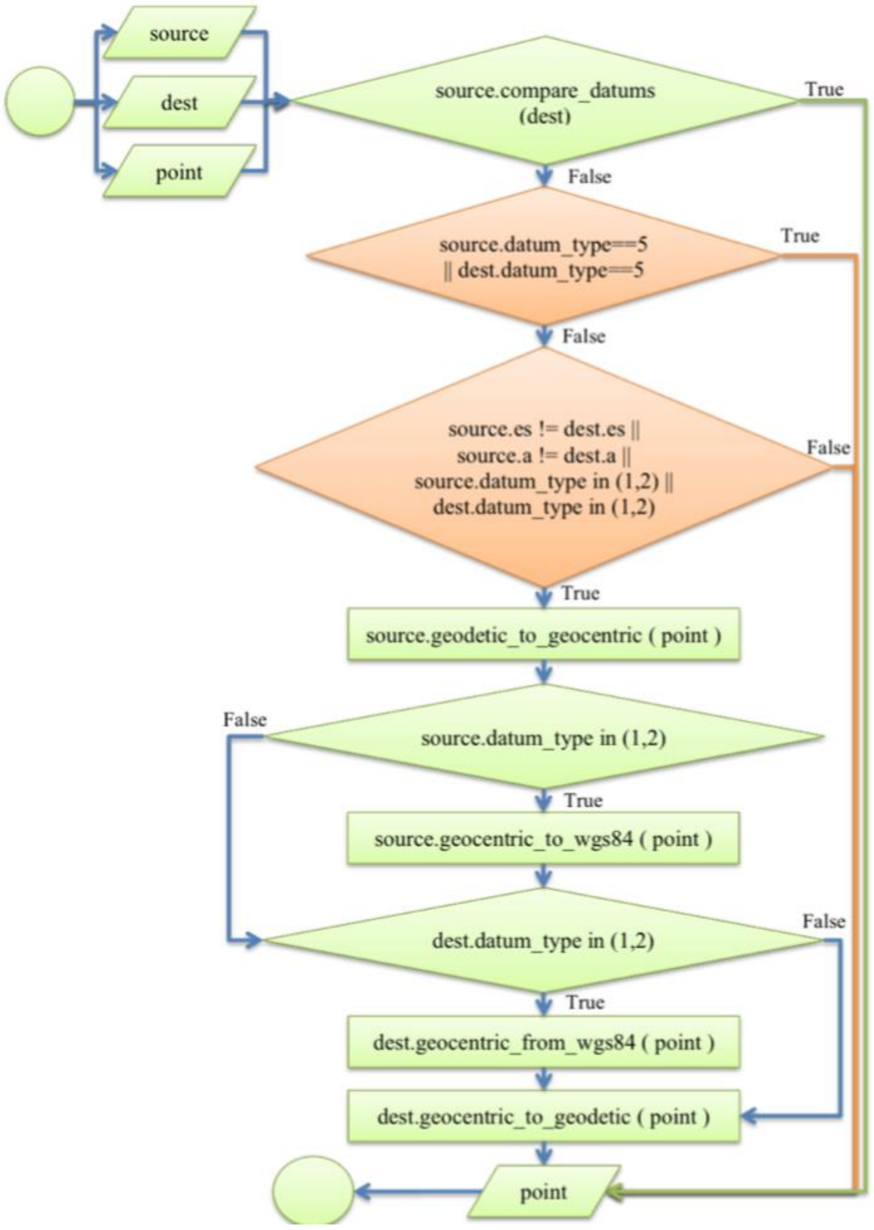
source
dest
point
source.compare_datums (dest)
True
False
source.datum_type==5 || dest.datum_type==5
True
False
source.es != dest.es || source.a != dest.a || source.datum_type in (1,2) || dest.datum_type in (1,2)
False
True
source.geodetic_to_geocentric (point)
False
source.datum_type in (1,2)
True
source.geocentric_to_wgs84 (point)
dest.datum_type in (1,2)
False
True
dest.geocentric_from_wgs84 (point)
dest.geocentric_to_geodetic (point)
point

Figure 8. Datum Transformation Algorithm

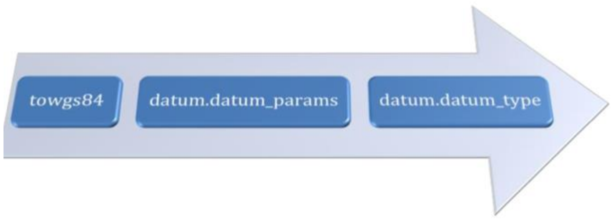

Figure 9. Determining *datum.datum_type* Map Projection Property

To involve datum transformation in the coordinate transformation, the *towgs84* property must be included into the Proj4JS format definition as can be seen in Listing 5. Based on the coordinate transformation algorithm as in Figure 7, if the Proj4JS format definition has *towgs84* property then the Proj4JS API transforms each coordinate into WGS84 map projection first before it transforms to the destination of the map projection as illustrated in Figure 10.

```
Proj4JS.defs["EPSG:3168"] = "+proj=omerc +lat_0=4
+lonc=102.25 +alpha=323.0257905
+lamda_c=323.1301023611111 +k=0.99984
+x_0=804670.24 +y_0=0 +a=6377295.664
+b=6356094.667915204 +units=m +towgs84=-11,851,5
+no_defs";
```

Listing 5. The Proj4JS Format Definition of "EPSG:3168" with *towgs84* Property

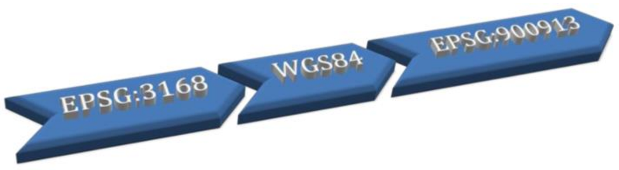

Figure 10. Coordinate Transformation Through WGS84 Map Projection

Once the *towgs84* property is added into the Proj4JS format definition of "EPSG:3168", the geographic features on the "EPSG:3168" layer projection are rendered on the same coordinate with the "EPSG:900913" layer projection as displayed in Figure 11. On this figure, the blue lines are rendered on top the green lines.

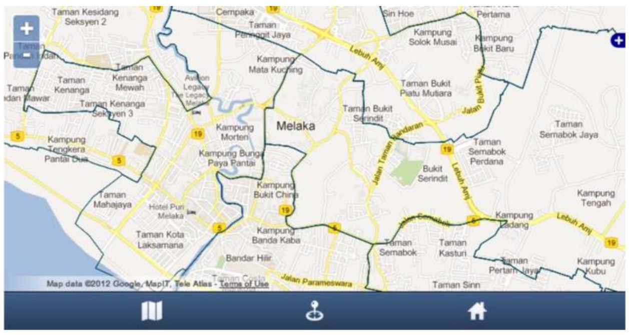

Figure 11. The Dun Melaka after Applying Datum Transformation

The same formula and Proj4JS format definition are used to render other GIS data that have more features, i.e. "Jalan Raya". This GIS data has 3953 features and they are rendered using "EPSG:3168" layer projection. The results are shown in Figure 12. The geographic features on this layer projection are rendered using red colour and the geographic features in the "EPSG:900913" layer projection is rendered using blue colour. In Figure 12, the red lines are rendered on top the blue lines. We also compare the result with the Google Map, and it shows that all lines are properly rendered.

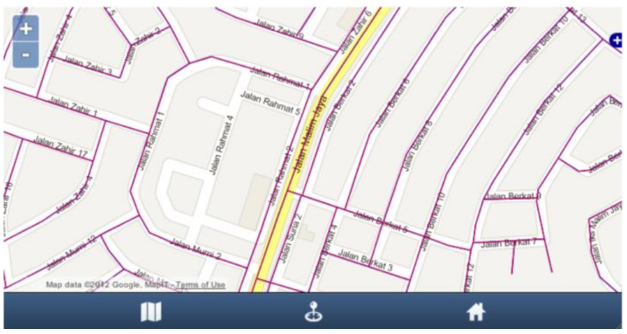

Figure 12. The Jalan Raya after Applying Datum Transformation

4. TESTING ON EPSG:3375

The same algorithm and formula, discussed above, are used to display other Hotine Oblique Mercator projection, i.e. "EPSG:3375". The Proj4JS format definition of "EPSG: 3375" is shown in Listing 6. The projection of data source is "EPSG:3375". The vector layer for "jalanraya_3375.json" is shown in Listing 7 and the results are shown in Figure 13.

```
Proj4JS.defs["EPSG:3375"] = "+proj=omerc +lat_0=4
+lonc=102.25 +alpha=323.0257964666666
+lamda_c=323.1301023611111 +k=0.99984 +x_0=804671
+y_0=0 +ellps=GRS80 +units=m +towgs84=-
11,851,5,0,0,0,0 +no_defs";
```

Listing 6. The Proj4JS Format Definition of "EPSG: 3375"

```
new OpenLayers.Layer.Vector("jalanraya_3375", {
projection: new OpenLayers.Projection("EPSG:3375"),
protocol: new OpenLayers.Protocol.URI({url:
"data/jalanraya_3375.json",
[...]
```

Listing 7. Vector Layer for "jalanraya_3375.json"

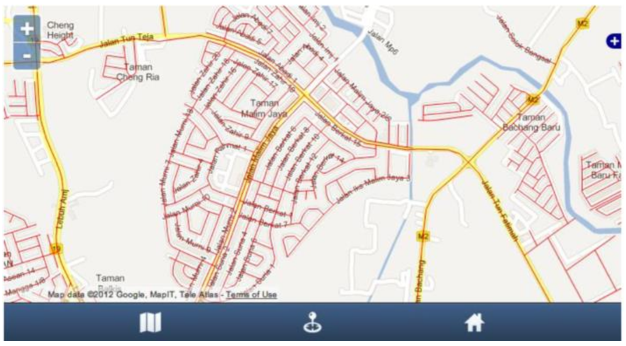

Figure 12. The Jalan Raya Using "EPSG: 3375" Layer Projection

5. CONCLUSION

The objective of this study is to enable the coordinate transformation from Hotine Oblique Mercator into another coordinate system. The result is a set of code and algorithm to handle the coordinate system transformation of Hotine Oblique Mercator projections applied into Proj4JS library.

The study was started on the map projection, coordinate system and coordinate transformation. Map projection is classified according to geometric and orientation of projection surface as can be seen in Figure 13 and the process involved in producing a map can be seen in Figure 14. Reference surface, known as reference ellipsoid, requires a model of the Earth to represent the Earth surface, scale reduction and map projection. It is a mathematical model of geometry shape to determine geodetic datum or set parameters such as equatorial plane radius or the semi-major axis, polar radius or the semi-minor axis, flattening, first and second eccentricity of the reference ellipsoid.

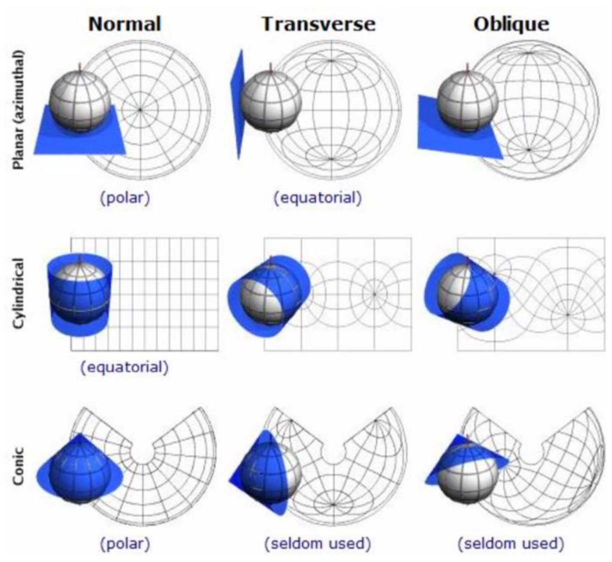

Figure 13. Map Projections with Different Geometric and Orientation of Projection Surface. (Source: Map Projections - Basic Definitions and Concepts, Carlos Alberto Furuti, 2012)

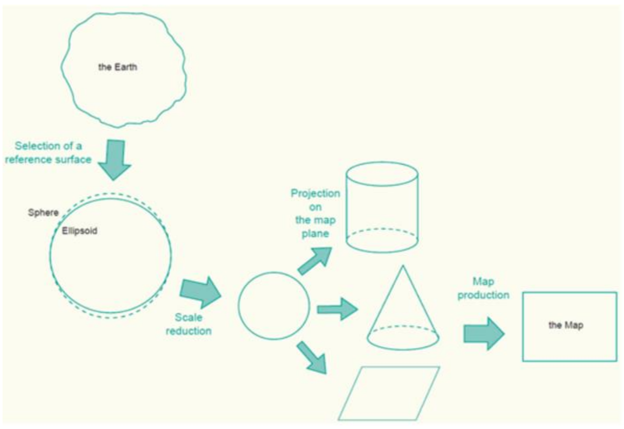

Figure 14. The Process of Representing the Earth on a Flat Map

(Source: Geometric Aspect of Mapping, R. Knippers, 2009)

To project a small local area with high accuracy into a plat projection surface, it requires a local reference ellipsoid with its geodetic datum that fits with the average sea level. “Note that the use of a different reference ellipsoid will result in a different latitude and longitude“ (R. Knippers, 2009). Local reference ellipsoid is used for a specific small area and global reference ellipsoid is used for Earth ellipsoid. Any local reference ellipsoid has a datum shift (position compared) to the global reference ellipsoid.

The coordinate transformation can be performed from the one coordinate system into the particular coordinate system. Moreover, two map projections may use different reference ellipsoids to produce a map for different areas on the earth surface. Since these maps use different reference ellipsoids, thus they use different geodetic datums too. Therefore, the transformation process should involve a datum transformation.

6. FUTURE WORK

This research is intended to bring GIS into the smartphone that implement a hybrid approach. Using hybrid application, the same code can work in different platform. Enabling the coordinate transformation on the client can support many hybrid applications to store the data locally. As a result, it supports an offline GIS application on the smartphone that can be used in disasters condition that has no internet connectivity.

ACKNOWLEDGEMENTS

This research was supported by the Ministry of Education, Malaysia and the Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka (UTeM). This research was funded through UTeM Short-Term Grant (PJP/2018/FTMK(6B)/S01634).

REFERENCES

Anshar K., Suryana N., 2014: Geospatial Content for Smartphone. *Adv. Sci. Lett. 20*, Numbers 10-12, October 2014, pp. 1793-1797(5), doi: 10.1166/asl.2014.5669

Anshar K., Suryana N., Othman Z., and Baharin S.S.K., 2015: Different Geospatial Data on Hybrid Map Application. *Journal of Networks*, Vol 10, No 7 (2015), 413-419, Aug 2015, doi:10.4304/jnw.10.7.413-419

OSGeo Foundation, 2012. Proj4JS User Guide. (http://trac.osgeo.org/Proj4JS/wiki/UserGuide)

OpenLayers, 2012: OpenLayers.Layer Constructor. (http://dev.openlayers.org/releases/OpenLayers-2.12/doc/apidocs/files/OpenLayers/Layer-js.html)

OGP, 2012: OGP Publication 373-7-2, Geomatics Guidance Note number 7, part 2 - Coordinate Conversions and Transformations including Formulas.

Furuti C.A., 2012: Map Projections - Basic Definitions and Concepts. (http://www.progonos.com/furuti/MapProj/Normal/CartDef/MapDef/mapDef.html)

Knippers R., 2009. Geometrics Aspect of Mapping. (http://plone.itc.nl/geometrics/Introduction/introduction.html)

International Hydrographic Bureau, 2008. User’s Handbook on Datum Transformations Involving WGS 84.

APPENDIX

APPENDIX A: “EPSG:3168” and “EPSG:900913” Map Projection Properties.

	Source	Destination
srsCode	EPSG:3168	EPSG:900913
datum.a	6377295.664	6378137
datum.es	0.00663784663019975	0
datum.datum_type	4	5
datum.datum_params	undefined	undefined
datumCode		
axis	enu	enu
projName	omerc	merc
to_meter		
from_greenwich		

APPENDIX B: Local Geodetic Datums and Transformation Parameters. (Source: User’s Handbook on Datum Transformations Involving WGS 84, International Hydrographic Bureau, 2008)

Continent: ASIA							
Local Geodetic Datums		Transformation Parameters					
Name	Code	ΔX(m)		ΔY(m)		ΔZ(m)	
INDONESIAN 1974 Indonesia	IDN	-24	±25	-15	±25	5	±25
KANDAWALA Sri Lanka	KAN	-97	±20	787	±20	86	±20
KERTAU 1948 West Malaysia and Singapore	KEA	-11	±10	851	±8	5	±6
KOREAN GEODETIC SYSTEM 1995 South Korea	KGS	0	±1	0	±1	0	±1