Facultad 6
Centro de Desarrollo GEySED
Departamento de GeoInformática

PROPUESTA DE ALGORITMOS PARA ANÁLISIS GEOMÉTRICOS EN SIG

TRABAJO FINAL PRESENTADO EN OPCIÓN AL TÍTULO DE
MÁSTER EN INFORMÁTICA APLICADA

Autor: Ing. Romanuel Ramón Antunez

Tutor: MSc. Yusnier Valle Martínez
Co-Tutor: MSc. Héctor Raúl González Diez

La Habana, Diciembre de 2011

DEDICATORIA

A mi esposa Lidisy.
A mis padres, Doris, Manuel y Bernardo.

DECLARACIÓN JURADA DE AUTORÍA

Declaro por este medio que yo Romanuel Ramón Antunez, con carné de identidad 84070426886, soy el autor principal del trabajo final de maestría “Propuesta de Algoritmos para Análisis Geométricos en SIG”, desarrollada como parte de la Maestría en Informática Aplicada y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Este trabajo fue desarrollado en el seno del proyecto GeneSIG y todos sus integrantes me reconocen la autoría principal del resultado expuesto.

Finalmente declaro que todo lo anteriormente expuesto se ajusta a la verdad, y asumo total responsabilidad moral y jurídica que se derive de este juramento profesional.

Y para que así conste, firmo la presente declaración jurada de autoría en La Habana a los ___ días del mes de _____ del año _____.

_____.

Nombre y Firma del maestrante

RESUMEN

La mayor parte de los análisis espaciales realizados en los Sistemas de Información Geográfica (SIG) hacen uso de cálculos geométricos sencillos, a partir de los cuales se construyen algoritmos más complejos.

Tradicionalmente estos cálculos se han desarrollado según métricas euclidianas, sin embargo, la información geográfica ha ganado en disponibilidad y exactitud planimétrica lo que ha provocado que estos cálculos realizados en los SIG sean poco precisos.

Estos errores pudieran llegar a ser notablemente importantes en algunos casos particulares, tales como en ámbitos muy extensos o proyecciones poco equidistantes. Por consiguiente, se hace necesario sustituir en aquellas herramientas SIG los clásicos cálculos euclidianos por otros cálculos que eliminen o disminuyan estos errores.

Para un correcto desempeño de estas funcionalidades es esencial analizar la calidad de las bases cartográficas que participan, las características del sistema de referencia (básicamente proyección y modelo elipsoide o esférico), las dimensiones del ámbito y sus objetivos (resolución de las bases resultantes, precisión planimétrica y temática requerida, entre otras). En función de estos parámetros, las herramientas de análisis SIG deberían facilitar al usuario no experto en estos temas una solución válida de forma automática, y al usuario avanzado un control adecuado para hacer valer su decisión de usar unos métodos, euclidianos, u otros, geodésicos. No sería muy apropiado obligar al usuario a usar siempre y en cualquier condición los métodos más precisos, pero mucho más lentos de la geodesia**.**

Por tal motivo en este trabajo se proponen un conjunto de algoritmos que permitan un correcto y eficiente funcionamiento de las funcionalidades para el cálculo de distancia, perímetro, acimut y área en aplicaciones SIG.

Palabras Claves: Algoritmo, Análisis Geométrico, Cálculo de Área de Polígonos, Geometría Computacional, SIG.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Gran parte de la información manejada a nivel empresarial y gubernamental en el mundo presenta una estrecha relación con datos espaciales. Por tal motivo la toma de decisiones y la precisión de estas están condicionadas, en gran medida por la calidad, exactitud y actualización de esta información espacial.

Se entiende por dato espacial todo aquel que tiene una referencia geográfica, de tal modo que se puede localizar "exactamente" donde sucede dentro de un mapa [Olaya, y otros, 2007].

Los Sistemas de Información Geográfica (SIG) son una tecnología que permite gestionar y analizar la información espacial y que surgió como resultado de la necesidad de disponer rápidamente de información para resolver problemas y contestar a preguntas de modo inmediato. Existen muchas definiciones de SIG, algunas de ellas acentúan su componente de base de datos, otras sus funcionalidades y otras enfatizan el hecho de ser una herramienta de apoyo en la toma de decisiones, pero todas coinciden en referirse a un SIG como un sistema integrado para trabajar con información espacial. Constituyen una herramienta esencial para el análisis y toma de decisiones en muchas áreas vitales para el desarrollo, incluyendo la relacionada con la infraestructura de un municipio, estado o incluso a nivel nacional o mundial [Peña Lloips, 2006].

Una de las disciplinas computacionales que presenta sus aportes de una forma bastante inmediata en estos sistemas es la Geometría Computacional.

Esta disciplina proporciona criterios para detectar y organizar estructuras geométricas, así como su representación en pantalla. Desarrolla herramientas computacionales para el análisis de problemas geométricos y propone estrategias para implementar algoritmos eficientes (en cuanto a tiempo de ejecución, preferiblemente igual o menor a la clase $O\,(nlogn)$) que faciliten la resolución efectiva de estos problemas computacionales [Coelho de Pina, 2004].

Generalmente en esta disciplina los problemas más tratados y aplicados en los SIG han sido trabajados según métricas euclidianas[1]. Por otro lado la información geográfica ha ganado en disponibilidad y exactitud planimétrica lo que ha provocado que estos cálculos realizados en los SIG mediantes métricas euclidianas sean poco precisos [Pesquer Mayos, y otros, 2003].

Estos errores pudieran llegar a ser notablemente importantes en algunos casos particulares, tales como en ámbitos muy extensos o proyecciones poco equidistantes. Por consiguiente, se hace necesario sustituir en aquellas herramientas SIG los clásicos cálculos euclidianos por otros cálculos que eliminen o disminuyan estos errores.

El desarrollo de la presente investigación tiene su génesis con la creación del proyecto de Investigación Desarrollo (I+D) llamado GeneSIG, proyecto llevado a cabo por la Universidad de Ciencias Informáticas (UCI), las FAR, específicamente el Centro de Desarrollo, Compatibilización y Integración de Soluciones Informáticas para la Defensa (UCID) y el grupo empresarial GEOCUBA.

Este proyecto tiene como propósito principal el desarrollo de la plataforma soberana homónima para el desarrollo de SIG en ambiente web. Esta plataforma cuenta con un conjunto de módulos donde se agrupan las mayorías de las funcionalidades de los SIG convencionales. Unos de estos módulos está orientado a las funciones de análisis básicos o cálculos geométricos, tales como cálculo de distancias, perímetros, áreas, entre otras.

Para un correcto desempeño de estas funcionalidades es esencial analizar la calidad de las bases cartográficas que participan, las características del sistema de referencia (básicamente proyección y modelo elipsoide o esférico), las dimensiones del ámbito y sus objetivos (resolución de las bases resultantes, precisión planimétrica y temática requerida, entre otras). En función de estos parámetros, las herramientas de análisis SIG deberían facilitar al usuario no experto en estos temas una solución válida de forma automática, y al usuario avanzado un control adecuado para hacer valer su decisión de usar unos métodos, euclidianos, u otros, geodésicos. No sería muy apropiado obligar al usuario a

[1] Referente al análisis geométrico en espacios donde se satisfacen los axiomas de Euclides de la geometría, también conocidos como espacios euclidianos.

usar siempre y en cualquier condición los métodos más precisos, pero mucho más lentos de la geodesia**.** [Pesquer Mayos, y otros, 2003].

Como resultado de la personalización de la Plataforma Soberana GeneSIG se obtienen sistemas adecuados a negocios específicos y de diversas índoles, por lo que los usuarios finales de estos sistemas no tienen por qué ser necesariamente expertos en el manejo de SIG y temas de geodesia.

Debido a la inexistencia en el equipo de trabajo de técnicas para dar una respuesta válida y de forma automática a usuarios poco expertos para las funcionalidades de cálculos geométricos; se presenta la siguiente interrogante ¿Cómo obtener un correcto y eficiente funcionamiento de las funcionalidades para cálculos geométricos en SIG para diferentes niveles de usuarios[2]?

El objeto sobre el que se enfoca el estudio, desde el punto de vista tanto teórico como práctico, para dar solución al problema planteado, consiste en los algoritmos para los cálculos geométricos aplicados en SIG y, específicamente en los algoritmos eficientes existentes para el cálculo de distancia, perímetro, acimut[3] y área aplicados en SIG, lo que sería el campo de acción de la investigación.

Con el propósito de darle una solución efectiva al problema, se plantea como objetivo general: diseñar e implementar un conjunto de algoritmos, que permitan un correcto y eficiente funcionamiento de las funcionalidades para el cálculo de distancia, perímetro, acimut y área en aplicaciones SIG.

Varios son los algoritmos euclidianos basados en modelos geodésicos que se aplican en la resolución del problema descrito. En el marco del presente trabajo, se proponen un conjunto de algoritmos para cálculos geométricos partiendo de la siguiente hipótesis:

[2] En este trabajo se refiere a los niveles de usuarios según sus conocimientos y/o experiencia en el trabajo con SIG y temas de cartografía y geodesia.
[3] Es el ángulo de una dirección contado en el sentido de las agujas del reloj a partir del norte geográfico. En la geodesia o la topografía geodésica, el acimut sirve para determinar la orientación de un sistema de triangulación.

Si se diseña e implementa computacionalmente una propuesta de algoritmos para el análisis geométrico eficiente en SIG teniendo en cuenta la proyección[4] y/o sistema de referencia [5] con que se trabaja, se podrán ofrecer resultados válidos para las funcionalidades de cálculo de distancia, perímetro, acimut y área en SIG para diferentes niveles de usuarios.

Como tareas de investigación se proponen las siguientes:

- Identificación de las variantes de solución existentes y tendencias a seguir en la solución del problema planteado, a partir de un estudio profundo del estado del arte que precede la realización del presente trabajo.
- Diseño de un algoritmos genérico para análisis geométricos en SIG que tenga en cuenta el sistema de coordenadas y/o proyección con la que se trabaja.
- Implementación, según el algoritmo genérico propuesto anteriormente, de los algoritmos para el cálculo de distancias, perímetros y acimut.
- Diseño e implementación de un algoritmo para determinar el área de un polígono con coordenadas elipsoidales.
- Prueba de los resultados de los algoritmos propuestos en la plataforma GeneSIG.

La contribución fundamental de este trabajo de forma general, consiste en la propuesta de un conjunto de algoritmos para cálculos geométricos apoyados en el diseño de un algoritmo genérico, que tenga en cuenta el sistema de coordenadas y/o proyección con la que se trabaja; y específicamente el desarrollo de un nuevo algoritmo basado en la geometría esférica[6] para determinar el área de polígonos esféricos sin necesidad de utilizar modelos geodésicos para su re-proyección y/o eliminando restricciones a la hora de la digitalización de dichos polígonos.

Para un mejor desempeño en la investigación se emplearon los siguientes métodos:

Análisis y síntesis: Este método fue utilizado para analizar la situación problémica y determinar posibles variantes de solución.

[4] Técnica utilizada para representar la forma de la tierra en un plano. Para mejor comprensión ver Capítulo 1, epígrafe 1.1.
[5] Referido al elipsoide o Datum empleado. Para mejor compresión ver Capítulo 1, epígrafe 1.1.
[6] Geometría basada en el modelo de una esfera. Para mejor compresión ver Capítulo 1, epígrafe 1.2.

Hipotético – deductivo: Permitió, a partir de la hipótesis, elaborar conclusiones acerca de la factibilidad de la utilización de un algoritmo genérico de análisis geométricos en Sistemas de Información Geográfica.

Criterio de experto: Para asumir determinados criterios que son imprescindibles para realizar el desarrollo de la propuesta y su aplicación en Sistemas de Información Geográfica.

El presente documento se encuentra dividido en tres capítulos. En el primero de ellos, Fundamentación Teórica, se introducen algunos temas elementales de cartografía y geodesia, se introducen algunas nociones de la geometría esférica y se realiza un análisis de las técnicas más empleadas para el análisis geométrico actualmente y los SIG.

El segundo capítulo está dedicado a la construcción y presentación de la solución propuesta. Se comienza diseñando el algoritmo genérico de análisis geométricos en SIG; continúa con la presentación de la propuesta para los algoritmos para el cálculo de distancia, perímetro y acimut, y culmina con la presentación del diseño del algoritmo para el cálculo de áreas.

En el tercer capítulo se presenta un análisis de la complejidad del algoritmo de cálculo de área y se muestran de forma gráfica y tabulada los resultados obtenidos durante el proceso de investigación y desarrollo llevado a cabo.

Además de un conjunto de conclusiones a las que se arriban una vez concluido el trabajo.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

Con el objetivo de facilitar la comprensión del alcance de la investigación, en el presente capítulo se introducen una serie de temas fundamentales asociados al problema planteado. Se explica qué son y para qué se usan las proyecciones cartográficas, se introduce una breve panorámica sobre la geometría esférica que servirá de base para dar solución al problema tratado; concluyéndose con un análisis de algunos de los métodos clásicos empleados en los SIG para tratar los cálculos geométricos.

1.1 Cartografía y Geodesia. Sistemas de Proyección.

La Tierra presenta una forma irregular denominada Geoide, al cual define [Ortiz, 2003] como figura irregular que trata de ajustarse a la forma de la tierra. El Geoide se puede definir como la superficie equipotencial definida por los mares en calma prolongados por debajo de los continentes. Equipotencial significa que en todos sus puntos la vertical astronómica -dirección que siguen los objetos que caen atraídos por la gravedad- es normal (perpendicular) al Geoide. Debido a las variaciones en los materiales que componen la superficie, la densidad de la tierra no es uniforme en todos sus puntos, y ello provoca que el Geoide tienda a ser más alto en las zonas continentales que en los océanos, presentándose suaves depresiones y abultamientos en varias regiones del globo.

Debido a esta irregularidad del Geoide, plantea [Fallas, 2003], se introducen ambigüedades en la localización de objetos en la superficie terrestre y por tanto en las mediciones. Es por tal motivo que se ha decidido utilizar una superficie de referencia abstracta que aproxime la forma del Geoide, pero sin sus irregularidades; esta figura se denomina esferoide/elipsoide.

Continua [Fallas, 2003] explicando que estos modelos se definen mediante dos parámetros, el tamaño del semieje mayor (a) y el tamaño del semieje menor (b). El achatamiento del esferoide se define entonces mediante un coeficiente como:

$$\alpha = {}^{1}/_{f}$$

$$f = {}^{(a-b)}/_{a}$$

Alterando los valores de los coeficientes a y b se obtienen diferentes esferoides/elipsoides. Se han propuesto diversos esferoides/elipsoides de referencia, generalmente se conocen con el nombre de su creador. La razón de tener diferentes esferoides/elipsoides es que ninguno de ellos puede adaptarse completamente a todas las irregularidades del Geoide, aunque cada uno de ellos se adapta razonablemente bien a una zona concreta de la superficie terrestre.

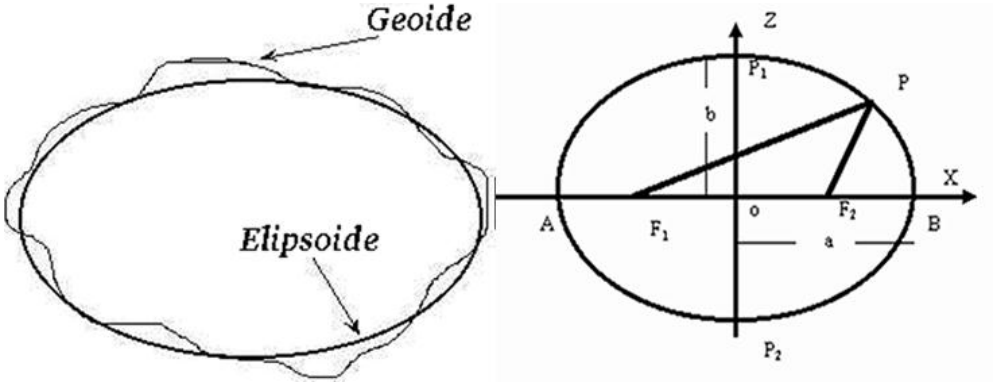

Figura 1. Esferoide/Elipsoide y Geoide.

Para obtener una mejor aproximación de este modelo al Geoide es necesario anclar el esferoide/elipsoide a un punto fundamental del Geoide, en que el esferoide/elipsoide y el Geoide son tangentes.

Surge el concepto de datum que es el conjunto formado por los parámetros a y b del elipsoide, las coordenadas geográficas, latitud y longitud (λ y ω), del punto fundamental y la dirección que define el Norte [Goizueta, 2006].

Una de las principales ventajas del esferoide/elipsoide es que puede ser modelado mediante ecuaciones matemáticas, y por ende su superficie puede ser representada en un plano en dos dimensiones, más conocido como mapa. El proceso de transformar las coordenadas del esferoide/elipsoide en coordenadas planas para su representación en dos dimensiones se conoce como proyección cartográfica y es el campo de estudio de la ciencia cartográfica.

El problema fundamental al proyectar el esferoide/elipsoide es que no existe modo alguno de representar en un plano toda su superficie sin deformarla, por lo que el objetivo va a ser minimizar, dentro de lo posible, estas deformaciones [Fallas, 2003].

Mediante una proyección cartográfica se hace corresponder cada punto del plano con uno del esferoide/elipsoide y viceversa. Por tal motivo se puede definir como una biyección que viene dada por ecuaciones de la forma:

$$x = f_1(\omega, \lambda)$$
$$y = f_2(\omega, \lambda)$$

Para obtener estas ecuaciones se proyecta la porción de la superficie terrestre que va a cartografiarse sobre una figura geométrica (cilindro, cono, o plano) que si puede transformarse a un plano sin distorsiones. De este modo se pueden clasificar las proyecciones en función del objeto geométrico utilizado para proyectar en cilíndricas, pseudocilíndricas, cónicas y acimutales o planas [Evenden, 2003].

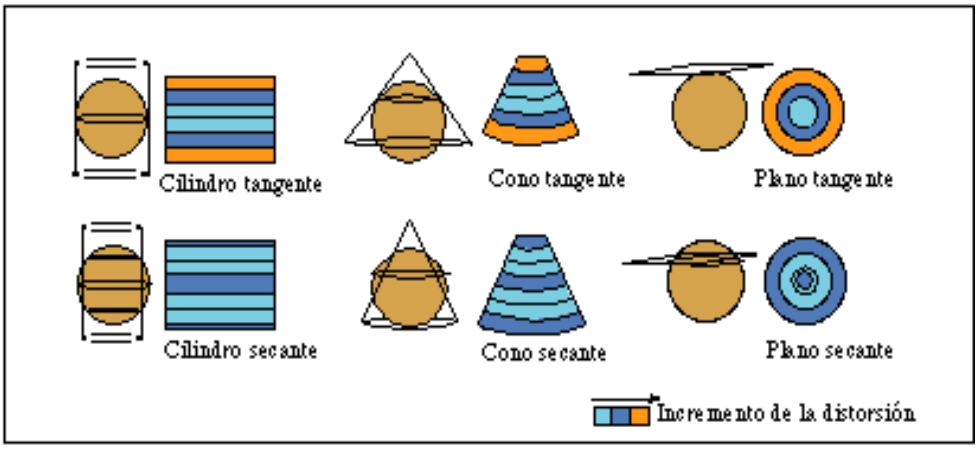

Figura 2. Tipos de Proyecciones.

Una proyección implica siempre una distorsión, el objetivo de la cartografía es minimizar estas distorsiones utilizando la técnica de proyección más adecuada en cada caso. Las propiedades del elipsoide que pueden mantenerse son:

Conformidad: si un mapa mantiene los ángulos que dos líneas forman en la superficie terrestre, se dice que la proyección es conforme. El requerimiento para que haya conformidad es que en el mapa los meridianos y los paralelos se corten en ángulo recto y que la escala sea la misma en todas las direcciones alrededor de un punto, sea el punto

que sea. Una proyección conforme mantiene además las formas de polígonos pequeños. Se trata de una propiedad fundamental en navegación [Goizueta, 2006].

La desventaja de las proyecciones de tipo conforme es que distorsionan fuertemente el tamaño de las superficies cartografiadas y como consecuencia la escala no es constante entre regiones del mapa. Por ejemplo, en un mapamundi las superficies en altas latitudes se muestran más grandes de lo que realmente son [Fallas, 2003].

Equivalencia: es la condición por la cual una superficie en el plano de proyección tiene la misma superficie que en la esfera. La equivalencia no es posible sin deformar considerablemente los ángulos originales, por lo tanto, ninguna proyección puede ser equivalente y conforme a la vez. Resulta conveniente por ejemplo en planos catastrales [Goizueta, 2006].

Equidistancia: cuando una proyección mantiene las distancias reales entre dos puntos situados sobre la superficie del Globo (representada por el arco de Círculo Máximo que las une) [Goizueta, 2006].

Dirección: en una proyección acimutal o cenital las direcciones o acimuts de todos los puntos son correctas respecto del centro de proyección. Existe una proyección acimutal que es equivalente, otra que es conforme y una tercera que es equidistante [Goizueta, 2006].

1.2 Nociones de Geometría Esférica.

Teniendo en cuenta que el modelo utilizado para la representación del geoide, es el esferoide/elipsoide, se procede a analizar elementos básicos de esta geometría.

La geometría esférica es el modelo más simple de la geometría elíptica, esta fue propuesta por Riemann. El modelo para esta geometría es una esfera, donde las rectas pasan a ser geodésicas, que no son más que círculos máximos de la esfera [García Alvarado, 2002].

Definición 1.2.1 Se llama *circunferencia máxima, ciclo* a la intersección de la superficie esférica con un plano que pase por su centro [S. Chinea, 2002].

Definición 1.2.2 Se llama *circunferencia mínima o menor* a aquella que se obtiene como intersección de la esfera con planos que no pasan por su centro [Barrero Ripoll, y otros, 2008].

Definición 1.2.3 Se denominan *polos de un ciclo* a los extremos del diámetro de la esfera que es perpendicular al plano que define el ciclo [Barrero Ripoll, y otros, 2008].

En esta geometría, cada recta a través de un punto *P* no contenido en la recta *AB* intercepta a la recta *AB*. En esta geometría es posible la no existencia de paralelas [García Alvarado, 2002].

Definición 1.2.4 Se denomina *distancia esférica* entre dos puntos como la longitud del menor arco del ciclo que los contiene [Iskategia, 2001].

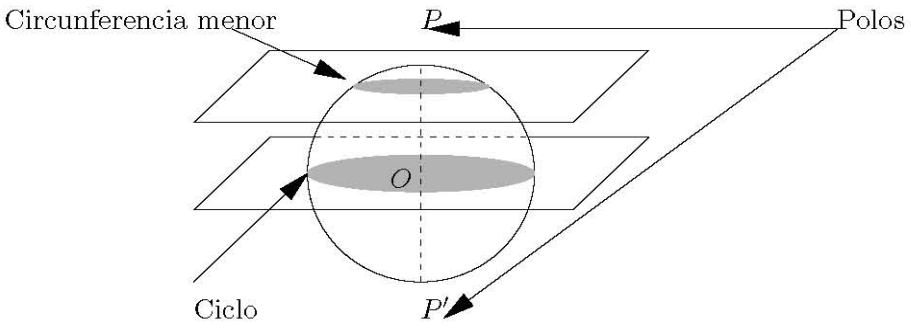

Figura 3.Ciclos, Polos y Circunferencias Menores.

Definición 1.2.5 Se denomina *ángulo esférico* entre dos ciclos al ángulo formado por las semitangentes a las circunferencias en uno de sus puntos de contacto. También se puede definir como el ángulo diedro que forman los planos que determinan los ciclos [S. Chinea, 2002].

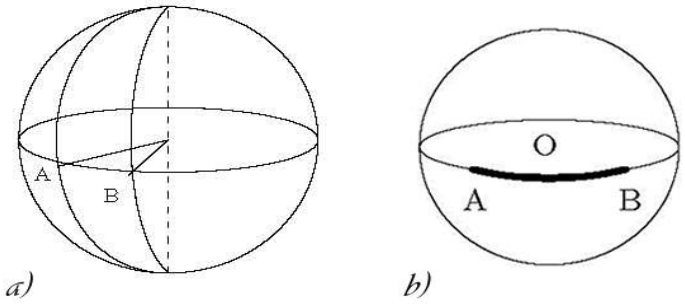

Figura 4. Ángulo Esférico y Distancia Esférica.

1.2.1 Triángulos Esféricos. Propiedades.

Definición 1.2.1.1 Se llama *triángulo esférico* a la porción de superficie esférica limitada por tres arcos de ciclo [S. Chinea, 2002].

Observación 1.2.1.1 Los lados de un triángulo esférico, si bien son arcos de ciclo, se considerarán como medidas angulares. En caso de querer conocer la medida de longitud del arco, bastará multiplicar por el radio de la esfera.

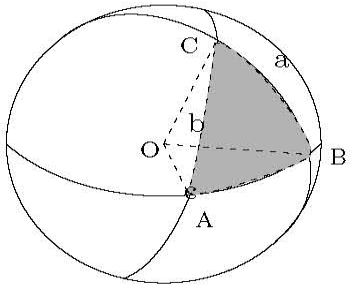

Figura 5. Triángulo Esférico

Los lados del triángulo esférico son los arcos *a; b* y *c*; los vértices o ángulos *A, B* y *C* son los ángulos diedros que forman los arcos dos a dos.

Definición 1.2.1.2 Se denomina *ángulo triedro* al formado por tres semirrectas en el espacio, no situadas en el mismo plano, con un vértice común V [Barrero Ripoll, y otros, 2008].

Teorema 1.2.1.1 A cada una de las propiedades de un triángulo esférico le corresponde una propiedad análoga de su ángulo triedro asociado [Iskategia, 2001].

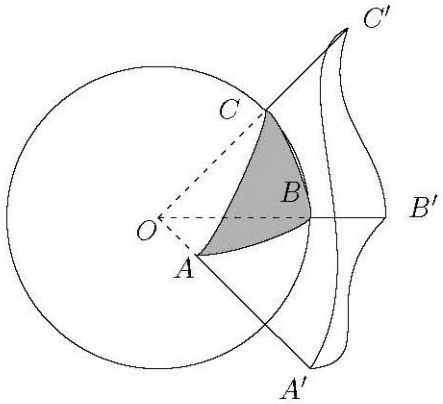

Figura 6. Ángulo Triedro asociado a un Triángulo Esférico

1.2.1.1 Propiedades.

1. Cualquier lado de un triángulo esférico es menor que una semicircunferencia $a < 180º$ [Iskategia, 2001].

2. Cada lado de un triángulo esférico es menor que la suma de los otros dos y mayor que el módulo de su resta [Iskategia, 2001].

$$|c - b| < a < c + b$$

3. La suma de los lados de un triángulo esférico es menor que cuatro rectos [Barrero Ripoll, y otros, 2008].

$$a + b + c < 360º$$

4. En un triángulo esférico a mayor lado se opone mayor ángulo y viceversa [Iskategia, 2001].

$$a < b \leftrightarrow A < B$$

5. En un triángulo esférico a lados iguales se oponen ángulos iguales y viceversa [Iskategia, 2001].

$$a = b \leftrightarrow A = B$$

6. La suma de los ángulos interiores en un triángulo esférico es mayor que dos rectos y menor que seis rectos [Barrero Ripoll, y otros, 2008].

$$180º < A + B + C < 540º$$

1.2.1.2 Área de un Triángulo Esférico.

Definición 1.2.1.2.1 Se denomina *exceso esférico* de un triángulo esférico *ABC* y se denota por ε al valor del ángulo en que la suma de los tres ángulos del triángulo esférico excede a ±180, es decir [Iskategia, 2001]:

$$\varepsilon = A + B + C - 180 \text{ ó } \varepsilon' = A + B + C - \pi$$

Según si los ángulos vienen expresados en grados o en radianes respectivamente.

Sea *ABC* un triángulo esférico sobre una esfera de radio *R*; se calcula su área con la fórmula [Barrero Ripoll, y otros, 2008]:

$$S = \frac{\pi R^2 \varepsilon}{180} \qquad o \qquad S = R^2 \varepsilon'$$

El cálculo del área de un triángulo esférico, es un caso particular de una ecuación más general definida para el cálculo del área de polígonos esféricos [Bevis, y otros, 1987].

$$S = \frac{\pi R^2}{180}(A_1 + A_2 + \cdots + A_n - (n-2) * 180)$$

En radianes quedaría:

$$S = R^2(A'_1 + A'_2 + \cdots + A'_n - (n-2) * \pi)$$

1.2.2 Elementos de Trigonometría Esférica.

Teorema 1.2.2.1 En un triángulo esférico, los senos de los lados son proporcionales a los senos de los ángulos opuestos [S. Chinea, 2002].

$$\frac{\sin a}{\sin A} = \frac{\sin b}{\sin B} = \frac{\sin c}{\sin C}$$

Esta relación permite calcular un lado o un ángulo, conocido su ángulo o lado opuesto y otro par de elementos opuestos. La dificultad radica en calcular el tercer ángulo, pues aquí no hay relación constante entre los ángulos de un triángulo. En este caso es útil la siguiente expresión.

Teorema 1.2.2.2 En todo triángulo esférico, el coseno de un lado es igual al producto de los cosenos de los otros dos lados, más el producto de los senos de dichos lados por el coseno del ángulo comprendido [S. Chinea, 2002].

$$\cos a = \cos b \cos c + \sin b \sin c \cos A$$

De forma análoga se puede obtener una ecuación de este tipo para los ángulos, obteniéndose las ecuaciones que relacionan tres ángulos y un lado:

$$\cos A = -\cos B \cos C + \sin B \sin C \cos a$$
$$\cos B = -\cos A \cos C + \sin A \sin C \cos b$$
$$\cos C = -\cos B \cos A + \sin B \sin A \cos c$$

A partir del teorema del seno y del coseno no es muy complejo deducir el de la cotangente, quedando planteado [S. Chinea, 2002]:

$$\cot a \sin b = \cos b \cos C + \sin C \cot A$$

1.3 Cálculos Geométricos en los SIG.

Según [Olaya, y otros, 2007] la mayor parte de los análisis espaciales realizados en los SIG hacen uso de cálculos geométricos sencillos, a partir de los cuales se construyen algoritmos más complejos.

Se plantea en [Alonso Sarría, 2006] que esto se debe a que prácticamente todos los cálculos geométricos que se realizan en estos sistemas (fundamentalmente los realizados sobre capas vectoriales[7]) se basan en la posición y las relaciones topológicas entre objetos.

1.3.1 Cálculos Geométricos según Métricas Euclidianas.

La idea de distancia es fundamental para todo análisis espacial, pues en casi todos los procedimientos se incluye este concepto. En el plano la distancia euclídea entre dos puntos viene dada por [Alonso Sarría, 2006]:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

En el análisis geográfico es habitual utilizar la denominada distancia de Manhattan[8] cuya expresión es:

$$d_m = (x_2 - x_1) + (y_2 - y_1)$$

Tanto la distancia euclídea como la de Manhattan son casos particulares de las métricas LP, que responden a la forma:

[7] En el formato vectorial los diferentes objetos se representan como puntos, líneas o polígonos.
[8] Se denomina así a que es similar a las recorridas por las calles dispuestas regularmente tales como las de la ciudad de Manhattan.

$$d^{\beta} = \left(\left| |x_2 - x_1| \right|^p + \left| |y_2 - y_1| \right|^p \right)^{\frac{\beta}{p}}$$

Para el caso $p = 1$ se obtiene la distancia de Manhattan y para $p = 2$ la distancia euclídea [Olaya, y otros, 2007].

Además de calcular la distancia entre dos puntos, puede calcularse entre geometrías. Es el caso entre 2 rectas en el plano es igual a la distancia entre un punto cualquiera sobre la primera hasta otro punto ortogonal a él en la segunda, siempre y cuando sean paralelas, en caso contrario la distancia es nula pues siempre existirá un punto en que ambas se encuentran [Olaya, y otros, 2007].

Sin embargo, en los SIG no suele trabajarse con rectas de longitud infinita en el sentido matemático, sino con segmentos de estas.

La distancia de un segmento definido por sus extremos (x_1, y_1) y (x_2, y_2) a un punto (x_3, y_3) se calcula como la distancia de este punto hasta el punto sobre el segmento en el que se intercepta la recta que pasa por el punto (x_3, y_3) y es perpendicular al segmento en cuestión [Alonso Sarría, 2006].

Este punto de intercepción, si existe (puede no existir por condiciones de paralelismo o no posean las longitudes necesarias para encontrarse), se puede obtener según [Olaya, y otros, 2007]:

$$x = x_1 + u(x_2 - x_1)$$
$$y = y_1 + u(y_2 - y_1)$$

Donde u se calcula de la forma:

$$u = \frac{(x_3 - x_1)(x_2 - x_1) + (y_3 - y_1)(y_2 - y_1)}{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

De esta manera se puede obtener igualmente la distancia de un punto a un polígono, siendo la distancia entre el punto y la línea que define el perímetro de dicho polígono.

Para el caso de los polígonos, son dos magnitudes principales las que se analizan, área y perímetro. Para el caso del área se calcula de la siguiente forma:

$$A = \left|\frac{1}{2}\sum_{i=1}^{n} x_i y_{i+1} - x_{i+1} y_i\right|$$

Para aquellos polígonos que contengan huecos basta con restar al área calculada, la correspondiente al hueco y esta se calcula de igual manera, pues los huecos se definen al igual que los polígonos, como una polilínea cerrada [Olaya, y otros, 2007].

El perímetro de un polígono no es más que la suma de las distancias entre vértices adyacentes, es decir [Alonso Sarría, 2006]:

$$P = \sum_{i=1}^{n} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$$

1.3.2 Cálculos Geodésicos.

Los cálculos presentados hasta el momento por [Alonso Sarría, 2006] y [Olaya, y otros, 2007] son referidos al plano, y es como tradicionalmente se han realizado en los SIG. Sin embargo, cuando se trabaja con un sistema de coordenadas diferente a las cartesianas estos resultados son completamente erróneos. En [Pesquer Mayos, y otros, 2003] se plantean tres algoritmos para solucionar estos problemas que pueden generar los cálculos euclidianos.

El **Problema Inverso de la Geodesia**, consiste en determinar el acimut y la longitud de la línea geodésica que separe dos puntos sobre el elipsoide. Existen múltiples métodos para su resolución, pero uno de los más empleados en la literatura científica es el método iterativo de Bessel, el cual se explica en [Zakatov, 1981]:

- Partiendo de un par de puntos de coordenadas $P_1(\varphi_1, \lambda_1)\ y\ P_{21}(\varphi_2, \lambda_2)$ se determinan las variables auxiliares:

$$tg(u_1) = \frac{b}{a} tg(\varphi_1) \qquad tg(u_1) = \frac{b}{a} tg(\varphi_1)$$

$$\Delta l = \lambda_1 - \lambda_2 \qquad mes = \frac{u_1 + u_2}{2} \qquad men = \frac{u_1 - u_2}{2}$$

- Se realiza el siguiente proceso iterativo que finaliza cuando la diferencia $\Delta l - \omega$ es menor que la precisión solicitada:

$$\omega_2 = \frac{\Delta l}{2} \qquad g_1 = \frac{\cos(men)}{\sin(mes)\tan(\omega_2)} \qquad g_2 = \frac{\sin(men)}{\cos(mes)\tan(\omega_2)}$$

$$a_1^2 = \tan^{-1}(g_1) + \tan^{-1}(g_2) \qquad a_2^2 = \tan^{-1}(g_2) - \tan^{-1}(g_1) + \pi$$

$$M = \tan^{-1}\left(\frac{\tan(u_1)}{\cos a_1^2}\right) \qquad m = \cos^{-1}\left(\frac{\sin(u_1)}{\sin(M)}\right)$$

$$\sigma = \cos^{-1}(\sin(u_1)\sin(u_2) + \cos(u_1)\cos(u_2)\cos(\Delta l))$$

$$t_1 = e^2 \sin(m)\sigma\left(0.5 + \frac{e^2}{8} - \frac{e^2\cos^2(m)}{8}\right)$$

$$t_2 = \frac{e^4 \sin(m)\cos^2(m)\sin(\sigma)\cos(2M+\sigma)}{16}$$

$$si\ \Delta l > 0\ \omega = \Delta l + t_1 + t_2 \qquad si\ \Delta l < 0\ \omega = \Delta l - t_1 - t_2$$

- Una vez iterado dicho proceso se determina la(s) distancia(s) entre los dos puntos:

$$s = Ab\sigma - Bb\sin(\sigma)\cos(2M+\sigma) - Cb\sin(2\sigma)\cos(4M+2\sigma)$$

$$A = 1 + \frac{k^2}{4} - 3\frac{k^4}{64} \qquad B = \frac{k^2}{4} - \frac{k^4}{16} \qquad C = \frac{k^4}{128} \qquad k = e'\cos(m)$$

El **Problema Directo de la Geodesia**, consiste en determinar un punto destino problema a partir de un punto origen, una distancia y un acimut conocidos. Para este también existen diversos métodos conocidos para su resolución siendo uno de los más utilizados la integración de Runge-Kutta de 4to orden, también explicado en [Zakatov, 1981]:

- Partiendo de un punto de coordenadas $P(\varphi, \lambda)$ y conocidas una distancia (S) y un acimut (Z), se calculan las siguientes variables auxiliares:

$$k_1 = h\varphi'(\varphi_n, Z_n) \quad l_1 = h\lambda'(\varphi_n, Z_n) \quad m_1 = hZ'(\varphi_n, Z_n) \quad h = {}^{S}/_{n}$$

$$k_2 = h\varphi'\left(\varphi_n + \frac{k_1}{2}, Z_n + \frac{m_1}{2}\right) \quad l_2 = h\lambda'\left(\varphi_n + \frac{k_1}{2}, Z_n + \frac{m_1}{2}\right)$$

$$m_2 = hZ'\left(\varphi_n + \frac{k_1}{2}, Z_n + \frac{m_1}{2}\right)$$

$$k_3 = h\varphi'\left(\varphi_n + \frac{k_2}{2}, Z_n + \frac{m_2}{2}\right) \quad l_3 = h\lambda'\left(\varphi_n + \frac{k_2}{2}, Z_n + \frac{m_2}{2}\right)$$

$$m_3 = hZ'\left(\varphi_n + \frac{k_2}{2}, Z_n + \frac{m_2}{2}\right)$$

$$k_4 = h\varphi'(\varphi_n + k_3, Z_n + m_3) \quad l_4 = h\lambda'(\varphi_n + k_3, Z_n + m_3)$$

$$m_4 = hZ'(\varphi_n + k_3, Z_n + m_3)$$

- Donde las siguientes funciones provienen de las ecuaciones diferenciales de la línea geodésica:

$$\varphi'(\varphi, Z) = \frac{\partial\varphi}{\partial s} = \frac{\cos(Z)}{M} \quad \lambda'(\varphi, Z) = \frac{\partial\lambda}{\partial s} = \frac{\sin(Z)}{N\cos(\varphi)}$$

$$Z'(\varphi, Z) = \frac{\partial Z}{\partial s} = \frac{\sin(Z)\tan(\varphi)}{N}$$

$$N = \frac{a^2}{\sqrt{a^2\cos^2(\varphi) + b^2\sin^2\varphi}} \qquad M = \frac{a(1-e^2)}{(1-e^2\sin^2(\varphi))^{1.5}}$$

- Las coordenadas de cada nuevo paso en función del paso anterior son:

$$\varphi_{n+1} = \varphi_n + {}^{(k_1 + k_2 + k_3)}/_{6}$$

$$\lambda_{n+1} = \lambda_n + {}^{(l_1 + l_2 + l_3)}/_{6}$$

$$Z_{n+1} = Z_n + {}^{(m_1 + m_2 + m_3)}/_{6}$$

Que serán definitivas cuando en una nueva iteración las variaciones sean inferiores a la precisión marcada.

El tercer algoritmo planteado por [Pesquer Mayos, y otros, 2003] es referido al cálculo de áreas; donde se propone transformar las coordenadas de los vértices del polígono a analizar a una proyección equivalente y se propone específicamente una Cilíndrica Equal-Área, donde luego se puede aplicar la ecuación planteada en [Olaya, y otros, 2007].

Para la transformación de las coordenadas se emplean las siguientes ecuaciones [Pesquer Mayos, y otros, 2003]:

$$X = ak_0(\lambda - \lambda_0) \qquad Y = aq/(2k_0)$$

$$k_0 = \frac{\cos(\phi_s)}{\sqrt{1 - e^2 \sin^2(\phi_s)}}$$

$$q = (1 - e^2)\left[\frac{\sin(\phi)}{1 - e^2 \sin^2(\phi)} - \left(\frac{1}{2e}\right)\ln\left(\frac{1 - e\sin(\phi)}{1 + e\sin(\phi)}\right)\right]$$

1.3.3 Cálculos geométricos sobre la esfera.

Además de los algoritmos propuestos en [Pesquer Mayos, y otros, 2003], otros autores han hecho referencia a métodos de naturaleza más simple para determinar la distancia entre dos puntos y el área de un polígono sobre una superficie esférica.

Entre los métodos para determinar la distancia entre dos puntos sobre la superficie terrestre se tiene el caso de la ley de cosenos para trigonometría esférica [Iglesias, y otros, 1996].

$$a = \sin(lat_1) * \sin(lat_2)$$

$$b = \cos(lat_1) * \cos(lat_2) * \cos(lon_1) * \cos(lon_2)$$

$$c = cos^{-1}(a + b)$$

$$d = R * c$$

Aunque esta fórmula es matemáticamente exacta, varios autores plantean que es poco fiable computacionalmente para distancias pequeñas, pues el arco coseno puede introducir considerables errores.

Otro método es el empleo de la fórmula en coordenadas polares para el modelo de tierra plana:

$$a = \frac{\pi}{2} - lat_1$$

$$b = \frac{\pi}{2} - lat_2$$

$$c = \sqrt{a^2 + b^2 - 2 * a * b * \cos(lon_2 - lon_1)}$$

$$d = R * c$$

Esta nueva fórmula proporciona errores máximos más pequeños que la aproximación pitagórica para latitudes elevadas y grandes distancias.

Una tercera propuesta para calcular la distancia es la fórmula de Haversine [Mencia, 2006] y [MTL, 2008]:

$$dlon = lon_2 - lon_1$$

$$dlat = lat_2 - lat_1$$

$$a = sin^2\left(\frac{dlat}{2}\right) + \cos(lat_1) * \cos(lat_2) * sin^2\left(\frac{dlon}{2}\right)$$

$$c = 2 * sin^{-1}(\min(1.0, \sqrt{a}))$$

$$d = R * c$$

La fórmula de Haversine proporciona resultados matemática y computacionalmente exactos. El resultado intermedio c, es la distancia en radianes sobre el gran círculo. Mientras la distancia d, estará en las mismas unidades que R.

La aplicación del mínimo protege de posibles errores de redondeo que podrían afectar la computación del arcoseno si los dos puntos son antipodales, en lados opuestos de la Tierra. Bajo estas condiciones, la fórmula de Haversine tiene resultados adversos, pero el error que puede ser de hasta dos kilómetros está en el contexto de una distancia de cerca de 20 000 kilómetros, es decir cuatro órdenes de diferencia.

Para el caso del cálculo del área de polígonos simples sobre la esfera, en [Bevis, y otros, 1987] se propone un algoritmo basado en la formulacion del área de polígonos esféricos presentado en el epígrafe 1.2.

$$S = \frac{\pi R^2}{180}(A_1 + A_2 + \cdots + A_n - (n-2) * 180)$$

En radianes quedaría:

$$S = R^2(A'_1 + A'_2 + \cdots + A'_n - (n-2) * \pi)$$

Dicha fórmula solo depende de la amplitud de los ángulos interiores del polígono, y es precisamente esta amplitud la que intenta calcular el algoritmo presentado por [Bevis, y otros, 1987]. Sin embargo para poder garantizar que se midan exactamente los ángulos interiores se impone una restricción en este algoritmo, y es la necesidad de que el polígono se haya digitalizado en sentido horario, en caso contrario deberían ordenarse los

vértices del polígono a analizar obteniéndose una cota superior de complejidad para el algoritmo de $nlogn$.

1.3.4 Aplicación de métodos geométricos en SIG actuales.

La mayoría de los sistemas de información geográfica libre actualmente disponibles para la comunidad, tales como QuantumGIS, posibilitan herramientas para cálculos geométricos básicos, principalmente en herramientas para el cálculo de distancias entre puntos y áreas de polígonos. Estas funcionalidades se basan en los métodos planteados en el epígrafe 1.3.1 por lo que al trabajar en coordenadas geográficas las distancias y áreas quedan expresadas en grados y grados cuadrados respectivamente, unidades que no son válidas para estas mediciones.

Otras herramientas libres como Postgis -extensión de soporte espacial para PostgreSQL- presentan algunas alternativas a este problema. Tal es el caso de las funciones de Postgis ST_Distance, para el cálculo de distancia cartesiana entre geometrías basado en el método para la distancia presentado en el epígrafe 1.3.1, ST_Distance_Sphere, para el cálculo de la distancia entre geometrías sobre la esfera basado en el método de Haversine, además de las funciones ST_Azimuth para determinar el acimut entre dos puntos y ST_Area para determinar el área de un polígono. Esta última función está basada en el método presentado en el epígrafe 1.3.1 para el área de polígonos en el plano; por lo que al trabajar con coordenadas geográficas debe utilizarse la función ST_Transform para proyectar estas coordenadas a una proyección equivalente [Refractions Research, 2008].

En el ámbito propietario la mayoría de los sistemas, como es el caso de MapInfo, sí proveen herramientas capaces de solucionar estos problemas; pues posibilitan funciones para el cálculo de la distancia tanto cartesiana como esférica, opción que puede escoger el usuario siempre que sea posible. Para el caso del cálculo del área de polígonos proceden de forma similar pues implementan funciones para el área en el plano y el área en la esfera. Pero, al ser aplicaciones de código cerrado no se puede tener certeza si se basan en algunos de los métodos planteados anteriormente en el epígrafe 1.3 o en métodos propios.

1.4 Conclusiones Parciales.

Una vez analizados los puntos tratados en el capítulo, se puede concluir que el análisis geométrico en los SIG no es un problema simple si se quieren obtener resultados válidos. Debe el usuario ser consciente de la naturaleza de los datos con los que está trabajando, qué tipo de proyección y algoritmos utilizar en dependencia de los resultados que se quieran obtener.

Por otra parte, se presenta la geometría esférica como una geometría alternativa, que permite simplificar la geometría elíptica manteniendo sus propiedades. Mediante esta aproximación es posible encontrar algoritmos sencillos, que faciliten dar una respuesta válida para los cálculos geométricos a usuarios poco expertos en cartografía y geodesia de forma automática.

CAPÍTULO 2

ALGORITMOS PROPUESTOS

El presente capítulo está dedicado a explicar detalladamente la propuesta de solución al problema tratado. Primeramente se diseña el algoritmo genérico para análisis geométricos en dependencia del sistema de coordenadas con el que se trabaje o en dependencia de las preferencias y conocimientos del usuario. A continuación, se presentan los algoritmos propuestos para el cálculo de distancia, perímetro y acimut mostrándose cómo quedarían sus implementaciones con el empleo del algoritmo genérico anteriormente presentado. Finalmente se presenta el diseño del algoritmo para determinar el área de un polígono con coordenadas elipsoidales, conformándose de esa forma toda la propuesta de solución.

2.1. Algoritmo Genérico.

Como se presentó en el capítulo anterior una proyección cartográfica se puede definir como una biyección; luego por teorema se tiene que si f es una función biyectiva entonces f^{-1} existe y es también biyectiva [W. Weisstein, 2011].

Partiendo de este teorema es evidente que si se trabaja con coordenadas proyectadas es posible obtener las coordenadas elipsoidales correspondientes en el elipsoide de referencia del cual se partió.

El algoritmo genérico propuesto se basa en esta propiedad y teniendo en cuenta que en un sistema de información geográfica siempre es conocida la proyección y/o sistema de coordenadas en el que se está trabajando, la idea que se presenta es la siguiente:

- Si se está trabajando en un sistema proyectado entonces se puede trabajar con algoritmos para coordenadas planas o algoritmos para coordenadas elipsoidales con una previa transformación de las coordenadas planas rectangulares a elipsoidales.
  - Esta decisión puede ser tomada por el usuario en caso de ser experto; por defecto se trabajaría convirtiendo las coordenadas a elipsoidales.
- En caso de no estar trabajando en un sistema proyectado entonces se trabajaría con los algoritmos para las coordenadas elipsoidales.

El algoritmo genérico en pseudocódigo quedaría de la siguiente forma:

Algoritmo Genérico

```
function X (x: type)
        if projection do
                if cartesian do
                        return CartesianX(x: type)
                end if
                return SphericalX(Transform(x: type))
        end if
        return SphericalX(x: type)
end function
```

La variable *projection* es un booleano que tomaría valor verdadero en caso de estar trabajándose con un mapa proyectado y falso en caso distinto, mientras la variable *cartesian* sería otro booleano que se inicializaría en falso para trabajar por defecto con el algoritmo para las coordenadas elipsoidales, pero que puede cambiar su valor a petición del usuario.

2.2. Algoritmo para el cálculo de distancias.

Una vez diseñado el algoritmo genérico para el análisis geométrico en SIG, se procede entonces a proponer y diseñar en consecuencia al mismo, los algoritmos para el cálculo de distancias, perímetros, acimut y áreas.

Partiendo de que el determinar la distancia entre dos puntos será vital para los demás algoritmos se comienza entonces por la propuesta para este problema particular.

Para el diseño del algoritmo teniendo en cuenta el algoritmo genérico anteriormente planteado se deben proponer dos métodos para determinar la distancia entre dos puntos; un primer método para trabajar con coordenadas planas y un segundo método para trabajar con coordenadas elipsoidales.

2.2.1. Método para coordenadas planas.

Para el caso de coordenadas planas el problema es de resolución trivial y fue presentado en el capítulo anterior donde se plantea que viene dada por la fórmula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Por lo que el algoritmo en pseudocódigo sería bastante simple como se muestra en el algoritmo uno:

Algorithm 1. Cálculo de distancia en el plano

```
function CartesianDistanceCalculation (_pto1, _pto2: Point)
        return sqrt( pow( (_pto2.x –  pto1.x), 2 ) +  pow( (_pto2.y –  pto1.y ), 2 ) )
end function
```

Donde la función **sqrt** es la encargada de devolver el valor de la raíz cuadrada del número que recibe como parámetro, y la función **pow** es la encargada de elevar el primer número que recibe como parámetro a la potencia indicada por el segundo parámetro.

2.2.2. Método para coordenadas Elipsoidales.

Para esta instancia del problema se vieron diversos métodos, tanto iterativos como directos en el capítulo anterior. Sin embargo, en esta investigación se hará mayor énfasis en los métodos directos, pues por su naturaleza suelen ser más eficientes en cuanto a tiempo de ejecución que los iterativos.

Luego de analizar las variantes presentadas en el capítulo anterior se propone emplear la fórmula de Haversine, debido a que presenta una mejor estabilidad computacional respecto a las otras, además de ser uno de los métodos más ampliamente usados y citados en la literatura especializada. En el algoritmo dos se muestra su pseudocódigo.

Algorithm 2. Cálculo de distancia sobre la esfera

```
function SphericalDistanceCalcutation (_pto1, _pto2: Point, _radian: bool)
        _radio ← 6 378 400
        _dlon ← _pto2.x - _pto1.x
        _dlat ← _pto2.y - _pto1.y
        _intcal ←  pow(sin(_dlat/2),2) + cos(_pto1.y) * cos(_pto2.y) * pow(sin(_dlon/2),2)
        _intd ←  2 * arcsin(min(1.0,sqrt(z)))
        if _radian do
                return _intd
        end if
        return _radio * _intd
end function
```

2.2.3. Pseudocódigo de algoritmo genérico para la distancia.

Finalmente ya se tienen los métodos tanto para trabajar con coordenadas planas como en coordenadas elipsoidales; por lo que se está en condiciones de presentar el diseño completo del algoritmo.

Algorithm 3. Algoritmo genérico para el cálculo de distancia

```
function CalculateDistance (_pto1, _pto2: Point)
        if projection do
                if cartesian do
                        return CartesianDistanceCalculation (_pto1, _pto2)
                end if
                return   SphericalDistanceCalcutation(Transform(_pto1),  Transform(_pto2), true)
        end if
        return SphericalDistanceCalculation(_pto1, _pto2, true)
end function
```

2.3. Algoritmo para el cálculo de perímetros.

Ya una vez confeccionados las propuestas y el diseño del algoritmo genérico para determinar la distancia entre dos puntos; es posible confeccionar un algoritmo similar para el perímetro de un polígono o la longitud de una polilínea.

La solución a este problema es bastante trivial partiendo de la propuesta anterior, pues solo sería iterar en la propuesta anterior cada dos vértices adyacentes de la figura geométrica analizada.

2.3.1. Pseudocódigo de algoritmo genérico para el perímetro

Algorithm 4. Algoritmo Genérico para el cálculo del perímetro

```
Procedure PolygonPerimeter ( _p: Polygon)
    if projection do
                if cartesian do
                        for i ← 0 to n do
                                perimeter ←  CartesianDistanceCalculation (_p [i], _p[i+1])
                        end for
                else
                   for i ← 0 to n do
                      perimeter ← SphericalDistanceCalcutation(Transform(_p[i]),  Transform(_p[i+1]), true)
                    end for
                end if
        else
                for i ← 0 to n do
```

```
                    perimeter ← SphericalDistanceCalcutation(_p[i]), _p[i+1]), true)
            end for
        end if
        return perimeter
End procedure
```

2.4. Algoritmo para el cálculo de acimut.

El acimut no es más que la dirección angular de un punto respecto a otro, denominado punto base, y a la dirección norte, medido en sentido horario. Es decir, un punto al norte del punto base tendría un acimut de $0°$ o $360°$, uno ubicado al este tendría 90 º y así sucesivamente.

Por la definición de acimut, es necesario entonces tres puntos para su cálculo, punto base u origen, punto destino u objetivo y el punto norte. Los dos primeros puntos serían parámetros para el algoritmo no así el tercero, pues el punto de referencia norte puede ser calculable a partir de los dos iniciales.

La idea que se sigue entonces en el algoritmo sería, primero dados los puntos origen y destino, formar el punto norte tomando la abscisa del punto origen y la ordenada del punto destino. Seguidamente con los tres se determinarían las distancias desde el origen al destino y al punto norte respectivamente. Y finalmente con estas distancias y mediante trigonometría se puede obtener el ángulo buscado.

2.4.1. Pseudocódigo de algoritmo genérico para el acimut.

Algorithm 5. Algoritmo Genérico para el cálculo del acimut

```
function CalculateAcimut (ptoO, ptoD: Point)
        ptoN.x ← ptoO.x
        ptoN.y ← ptoD.y
        if projection do
                if cartesian do
                        distD ← CartesianDistanceCalculation (ptoO,ptoD)
                        distN ← CartesianDistanceCalculation (ptoO,ptoN)
                else
                        distD ← SphericalDistanceCalcutation(Transform(ptoO),
                        Transform(ptoD), true)
                        distN ← SphericalDistanceCalcutation(Transform(ptoO),
                        Transform(ptoN), true)
                end if
        else
```

```
                distD ← SphericalDistanceCalcutation(_p[i]), _p[i+1]), true)
                distN ← SphericalDistanceCalcutation(_p[i]), _p[i+1]), true)
        end if
        if distD > 0 do
                acimut ← arcocos(distN / distD)
                acimut ← acimut * 180 / Pi
        else
                acimut ← 0
        end if
        if ptoO.x ≤ ptoD.x  && ptoO.y ≤ ptoD.y do
                acimut = acimut
        else if ptoO.x > ptoD.x  && ptoO.y ≤ ptoD.y do
                acimut ← 360 - acimut
        else if ptoO.x ≥ ptoD.x  && ptoO.y > ptoD.y do
                acimut ← 180 + acimut
        else
                acimut ← 180 - acimut
        end if
        return acimut
end function
```

2.5. Algoritmo para el cálculo de áreas.

Para completar la propuesta de solución solo resta diseñar el algoritmo genérico para determinar el área de un polígono simple. Para ello primero se presenta el algoritmo para el caso de polígonos planos, se prosigue con el diseño de un algoritmo para el caso de polígono esféricos y finalmente se presenta el diseño general integrando ambos en el algoritmo genérico propuesto al inicio del capítulo.

2.5.1. Algoritmo para el área de polígonos simples en el plano.

Uno de los algoritmos más difundidos para hallar el área de un polígono en el plano está basado en la técnica incremental. Consiste en a partir de un vértice (vértice inicial) trazar diagonales a los restantes vértices, obteniendo una triangulación de dicho polígono, posteriormente el área del polígono en cuestión sería la sumatoria de las áreas de los triángulos resultantes.

Hallar el área de un triángulo en el plano, es un problema de resolución trivial, pues estaría dada por el producto vectorial dividido entre dos de $\overrightarrow{p_1p_2}$ y $\overrightarrow{p_2p_3}$, siendo p_1, p_2 y p_3 los vértices del triángulo, por lo que solo quedaría resolver el determinante de la matriz y dividir el resultado [Chen, 1996].

$$\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix}$$

Esto daría el área resultante con signo positivo si los vértices se toman en sentido anti horario y expresada en correspondencia con las unidad de medida en que estén expresados p_1, p_2 y p_3.

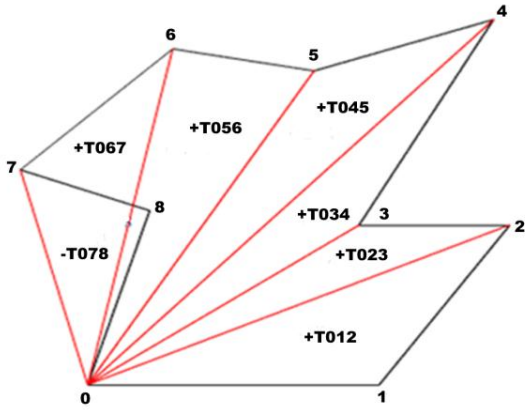

Figura 7. Obtención de triángulos para hallar el área del polígono.

Este algoritmo presenta una complejidad lineal $O(n)$ como se puede apreciar en el pseudocódigo del algoritmo seis:

Algorithm 6. Algoritmo para el cálculo del área de un polígono en el plano

```
function SurfaceCartesianPolygon (p: Polygon)
        area ← 0
        for i ← 0 to p.vertex – 2
                area ← area + LeftTurn(p.vertex[0], p.vertex[i], p.vertex[i+1])
        end for
        if area < 0 do
                return (area * -1) / 2
        end if
        return area / 2
end function

function LeftTurn( pto1, pto2, pto3: Point)
        return (pto1.x * pto2.y) – (pto1.y * pto2.x) + (pto2.x * pto3.y) – (pto2.y * pto3.x) +      (pto1.x * pto3.y)
        – (pto1.y *pto3.x)
end function
```

En el algoritmo presentado se calculan todas las áreas de los triángulos dobles y solo se divide al final, para de esta forma disminuir los errores por truncamiento que pueden existir al realizar los cálculos en un ordenador debido al redondeo.

2.5.2. Algoritmo para el área de polígonos simples esféricos.

Para el caso de polígonos simples sobre la superficie de la esfera se presentó en el capítulo anterior la propuesta de [Bevis, y otros, 1987], explicándose la restricción implícita en el mismo a la hora de digitalizar los polígonos y la cota superior de complejidad de no tenerse en cuenta dicha restricción.

Por tal motivo se diseña otra variante para determinar el área de estos polígonos eliminando esta restricción en la digitalización y de forma computacional menos costosa. El nuevo diseño mantiene la misma filosofía incremental del caso para polígonos en el plano con modificaciones en la forma de calcular el área de los triángulos.

El algoritmo tiene como única entrada el polígono al cual se le quiere determinar el área. Primeramente se selecciona un vértice como inicial, y a partir de este se trazan líneas geodésicas a los vértices restantes. De esta forma se obtienen un conjunto de triángulos esféricos. El próximo paso sería sumar o restar el área de cada uno de los estos triángulos en dependencia del giro realizado al recorrer sus vértices.

Para realizar el cálculo del área de estos triángulos, se calculan las longitudes de sus lados mediante Haversine. Luego por la fórmula del coseno para geometrías esféricas se hallan cada uno de los ángulos y finalmente con estos valores se puede obtener el área de dichos triángulos.

Algorithm 7. Algoritmo para el cálculo del área de un polígono sobre la esfera

```
function SurfaceSphericalPolygon (p: Polygon)
        area ← 0
        for i ← 0 to p.vertex – 2
                area← area + SurfaceSphericalTriangle(p.vertex[0],p.vertex[i],p.vertex[i+1])
        end for
        if area < 0 do
                return area * -1
        end if
        return area
end function
```

```
function SurfaceSphericalTriangle (pto1, pto2, pto3: Point)
        radio ← 6 378 100
        l1 ← SphericalDistanceCalcutation(pto1,pto2,false)
        l2 ← SphericalDistanceCalcutation(pto2,pto3,false)
        l3 ← SphericalDistanceCalcutation(pto1,pto3,false)
        a1 ← Fcoseno(l1,l2,l3)
        a2 ← Fcoseno(l2,l3,l1)
        a3 ← Fcoseno(l3,l1,l2)
        epsilon ← (a1 + a2 + a3) – Pi
        area ← epsilon * pow(radio, 2)
        if LeftTurn(pto1, pto2, pto3) < 0 do
                area ← area * -1
        end if
        return area
end function

function Fcoseno (l1, l2, l3: double)
        cose ← (cos(l1) – cos(l2) * cos(l3)) / (sin(l2) * sin(l3))
        return arccos(cose)
end function
```

De esta forma se mantiene el mismo orden de complejidad –lineal- que para el caso de polígonos planos.

2.5.3. Pseudocódigo de algoritmo genérico para el área.

Ya una vez que se cuenta con los algoritmos para el área de polígonos, tanto planos como esféricos, se puede pasar al diseño genérico vinculando ambos algoritmos. Inicialmente el algoritmo en pseudocódigo quedaría como sigue en el algoritmo ocho:

Algorithm 8. Variante 1 del algoritmo genérico para el cálculo del área de un polígono

```
function  CalculateSurface(p: Polygon)
        if projection do
                if cartesian do
                        return SurfaceCartesianPolygon(p)
                end if
                return SurfaceSphericalPolygon(Transform(p))
        end if
        return SurfaceSphericalPolygon(p)
end function
```

Sin embargo, como se puede apreciar en el algoritmo siete se emplea la función arco coseno y anteriormente se había comentado (en el subepígrafe 1.3.3) que esta función

podría introducir considerables errores para pequeñas distancias debido al comportamiento del coseno cerca del origen de coordenadas (Tabla 1).

Tabla 1. Valores de la función coseno

$\cos(x)$	Valor
5 grados	0.996194698
1 grado	0.999847695
1 minuto	0.9999999577
1 segundo	0.9999999999882
0.05 segundo	0.999999999999971

Por tal motivo y luego de una serie de corridas del algoritmo se pudo determinar que para áreas relativamente pequeñas -1000 hectáreas o 10 km^2- el algoritmo presentado hasta el momento no es fiable. Aunque el objetivo inicial que se perseguía era precisamente para polígonos relativamente grandes (mayores de 1000 hectáreas) se procede a dar solución a este inconveniente.

Estos casos particulares, para interés de este trabajo y en esta instancia del mismo, son tratados como polígonos planos. Salvedad que se asume debido al tamaño de dichos polígonos en relación a las dimensiones de la tierra, que tiene un radio aproximado de 6378.1 kilómetros, por lo que las mayores líneas geodésicas trazadas sobre ella tendrían una longitud de 40 054.468 kilómetros aproximadamente. Es sencillo intuir que la curvatura que presentan los lados que forman estos polígonos relativamente pequeños es mínima respecto a toda la geodésica.

A pesar de considerar estos polígonos como planos no es posible aplicar en ellos el algoritmo seis pues debido a que las coordenadas de dichos polígonos estarían expresadas en grados el área resultante quedaría expresada en grados cuadrados, unidad que no es válida para esta medición.
Para esta instancia del problema se propone entonces utilizar la fórmula de Herón para el área del triángulo la cual solo depende de la distancia de los lados del mismo. Quedando entonces el algoritmo en pseudocódigo:

Algorithm 9. Algoritmo para el cálculo del área de un polígono relativamente pequeño sobre la esfera

```
function SurfacePolygonHibridHeronSpherical(p: Polygon)
        area ← 0
        for i ← 0 to p.vertex – 2
                area ← area + SurfaceTriangleHeron(p.vertex[0],p.vertex[i],p.vertex[i+1])
        end for
        if area < 0 do
                return area * -1
        end if
        return area
end function

function SurfaceTriangleHeron(pto1, pto2, pto3: Point)
        l1 ← SphericalDistanceCalcutation(pto1,pto2,true)
        l2 ← SphericalDistanceCalcutation(pto2,pto3,true)
        l3 ← SphericalDistanceCalcutation(pto1,pto3,true)
        p ← (l1 + l2 + l3) /2
        int ← p * (p – l1) * (p – l2) * (p – l3)
        return sqrt(int)
end function
```

Ya una vez que se cuenta con el algoritmo para darle tratamiento a los polígonos como planos, solo falta definir a qué polígonos se les daría dicho tratamiento. Para esto se determina el rectángulo mínimo que recubre al polígono analizado; se determinaría el área de dicho rectángulo -área máxima esperada- y si esta es menor que 1000 hectáreas se trataría como plano, en caso contrario se aplicaría el procedimiento SurfaceSphericalPolygon. Quedando entonces finalmente el algoritmo genérico como se muestra a continuación, completándose de esta forma la propuesta de solución:

Algorithm 10. Algoritmo genérico para el cálculo del área de un polígono

```
function  CalculateSurface(p: Polygon)
        if projection do
                if cartesian do
                        return SurfaceCartesianPolygon(p)
                end if
                rectBox ← GetBox(p)
                pto1 ← rectBox.pto1
                pto2 ← rectBox.pto2
                pto3 ← rectBox.pto3
                l1 ← SphericalDistanceCalcutation(pto1,pto2,true)
                l2 ← SphericalDistanceCalcutation(pto2,pto3,true)
                areaExp ← l1 * l2
```

```
                if areaExp ≥ 10 000 000 do
                        return SurfaceSphericalPolygon(Transform(p))
                end if
                return SurfacePolygonHibridHeronSpherical(Transform(p))
        end if
        rectBox ← GetBox(p)
        pto1 ← rectBox.pto1
        pto2 ← rectBox.pto2
        pto3 ← rectBox.pto3
        l1 ← SphericalDistanceCalcutation(pto1,pto2,true)
        l2 ← SphericalDistanceCalcutation(pto2,pto3,true)
        areaExp ← l1 * l2
        if areaExp ≥ 10 000 000 do
                return SurfaceSphericalPolygon(p)
        end if
        return SurfacePolygonHibridHeronSpherical(p)
end function

function GetBox(p:Polygon)
        xmin ← p.vertex[0].x
        ymin ← p.vertex[0].y
        xmax ← p.vertex[0].x
        ymax ← p.vertex[0].y
        for i ← 0 to p.vertex
                if xmin > p.vertex[i].x do
                        xmin ← p.vertex[i].x
                end if
                if xmax < p.vertex[i].x do
                        xmax ← p.vertex[i].x
                end if
                if ymin > p.vertex[i].y do
                        ymin ← p.vertex[i].y
                end if
                if ymax < p.vertex[i].y do
                        ymax ← p.vertex[i].y
                end if
        end for
        rectBox ← new RectBox(xmin, ymin, xmax, ymax)
        return rectBox
end function
```

2.6. Conclusiones Parciales.

El algoritmo genérico de análisis geométrico propuesto, permite darles a usuarios avanzados la opción de elegir con qué tipo de método desea trabajar. Y a usuarios inexpertos en temas de cartografía y geodesia les garantiza obtener resultados válidos independientes del sistema de coordenadas con el que trabaje. Por tal motivo se propone que se trabaje por defecto con métodos para coordenadas geográficas, pues como se comentó en el capítulo anterior, las proyecciones cartográficas se construyen con diversos objetivos y si se empleara para otro diferente; los errores introducidos en los resultados serán mayores que los provocados en las aproximaciones obtenidas por los algoritmos para coordenadas geográficas.

Por lo explicado anteriormente esta propuesta permite mejorar el desempeño de las funcionalidades actuales de este tipo en sistemas como QuantumGIS, que no provee funciones geométricas para el trabajo con coordenadas geográficas. Además, ampliar la gama de funciones de Postgis, posibilitando que el usuario no necesite conocer o ser especialista en geodesia para poder transformar las coordenadas a una proyección válida, para poder obtener resultados correctos.

Por otro lado la utilización de las herramientas provistas por la geometría esférica permite obtener el diseño de un nuevo algoritmo para el cálculo del área de polígonos eliminando la restricción -a la hora de su digitalización- o posterior ordenamiento topológico de sus vértices, impuestos en el algoritmos presentado en [Bevis, y otros, 1987]; ganando de esta forma en simplicidad y tiempo.

CAPÍTULO 3

ANÁLISIS DE RESULTADOS

Una vez elaborada la propuesta en su totalidad se procede a analizar los resultados obtenidos con la misma y es este el objetivo del presente capítulo. Para realizar las pruebas, la propuesta fue implementada en la Plataforma Soberana GeneSIG, que tiene como objetivo facilitar el desarrollo de sistemas de información geográfica en entornos web.

Estas pruebas se centran principalmente en el algoritmo genérico para el cálculo del área de un polígono debido a que es el aporte particular del trabajo y en él se utiliza el algoritmo propuesto para determinar la distancia entre dos puntos. Algoritmo que es básico para los demás propuestos.

Primeramente se presenta el análisis de la complejidad del algoritmo para el área y luego se plasman una serie de resultados obtenidos tras varias corridas del algoritmo para diferentes polígonos.

3.1. Análisis de la complejidad del algoritmo para el área.

En el algoritmo genérico propuesto para el cálculo del área se llevan a cabo dos comparaciones de complejidad $O(1)$ y en dependencia del resultado se ejecutan funciones auxiliares que serían los que dicten la complejidad del algoritmo en general. Por lo que entonces se procede a analizar la complejidad de cada una de estas funciones.

3.1.1 Análisis de la complejidad de la función SurfaceCartesianPolygon.

La función *SurfaceCartesianPolygon* implementa un ciclo que repite $n-2$ veces la función *LeftTurn*, y en esta función a su vez solo se ejecutan operaciones elementales de suma, resta y multiplicación, por lo que tiene una complejidad $O(1)$. Teniéndose entonces el siguiente teorema.

Teorema 3.1.1.1: El área de un polígono en el plano puede ser computado con una complejidad $O(n)$. Dónde n es la cantidad de vértices del polígono.

3.1.2 Análisis de la complejidad de la función GetBox.

La función *GetBox* también implementa un ciclo que ejecuta n veces un conjunto de comparaciones de complejidad $O(1)$ y asignaciones de complejidad $O(1)$. Resulta entonces el siguiente lema:

Lema 3.1.2.1: El menor rectángulo que contiene a un polígono puede ser computado con una complejidad $\Theta(n)$. Dónde n es la cantidad de vértices del polígono.

3.1.3 Análisis de la complejidad de la función SphericalDistanceCalcutation.

Como se puede apreciar la función *SphericalDistanceCalcution* no implementa ciclo y sí solo un conjunto de asignaciones y cálculos matemáticos, por lo que su complejidad se puede decir a priori que sería $O(1)$. Sin embargo, se debe analizar el caso de las funciones raíz cuadrada y trigonométrica que se emplean en dicha función.

En [Brent, 1976], se plantea que todas estas funciones pueden ser calculadas en tiempo constante en dependencia del nivel de precisión que se quiera obtener. Llegándose entonces al lema:

Lema 3.1.3.1: La distancia entre dos puntos sobre la esfera puede ser computada con una complejidad $O(h)$. Dónde h es una constante que varía en dependencia de la precisión que se quiere obtener.

3.1.4 Análisis de la complejidad de la función SurfaceSphericalPolygon.

En la función *SurfaceSphericalPolygon* se sigue el mismo diseño que el planteado en la función *SurfaceCartesianPolygon*, por tal motivo a priori se podría decir que esta función presenta una complejidad $O(n)$ por el Teorema 3.1.1.1.

Sin embargo, en esta función se hacen uso de funciones trigonométricas por lo que esta cota sufre modificaciones por el lema 3.1.3.1 quedando entonces:

Teorema 3.1.4.1: El área de un polígono esférico puede ser computado con una complejidad $O(nh)$.

3.1.5 Análisis de la complejidad de la función SurfacePolygonHibridHeronSpherical.

En el caso de la función *SurfacePolygonHibridSpherical* sucede algo similar a la función *SurfaceSphericalPolygon* por lo que su complejidad es $O(nh)$.

3.1.6 Análisis de la complejidad del algoritmo genérico para el área.

Ya una vez analizada la complejidad de las funciones auxiliares utilizadas en la confección del algoritmo genérico para el cálculo del área (Algorithm 10) se puede proceder a analizar la complejidad del mismo.

Para el caso en que se esté trabajando con coordenadas proyectadas y se quiera utilizar un método para el plano, se tendría una complejidad $O(n)$ –por el Teorema 3.1.1.1- y en caso de que se quiera usar un método para coordenadas elipsoidales se tendría una complejidad $O(nh)$.

Para el caso en que se trabaje directamente con coordenadas elipsoidales se tendría una complejidad $O(nh)$.

Finalmente se tiene que:

Teorema 3.1.6.1: El área de un polígono se puede computar, teniendo en cuenta el sistema de coordenadas en que se encuentra, con una complejidad lineal.

3.2. Pruebas

Una vez analizada la complejidad del algoritmo genérico propuesto para el área se proceden a realizar un conjunto de pruebas para ver el comportamiento real del mismo en la Plataforma Soberana GeneSIG, o sea que tan próximo están los resultados obtenidos con la propuesta de los resultados arrojados con otras herramientas.

Estas pruebas fueron realizadas con geometrías en diferentes escalas, pues como bien se plantea en [Olaya, y otros, 2007], la información geográfica puede analizarse a distintos niveles y en dependencia del nivel empleado, los resultados pudieran ser de naturaleza diferentes.

Un ejemplo de este condicionamiento de la escala de análisis en los resultados obtenidos se puede apreciar al calcular el perímetro de una polilínea variando la dimensión de la medida empleada como se puede apreciar en la Figura 8.

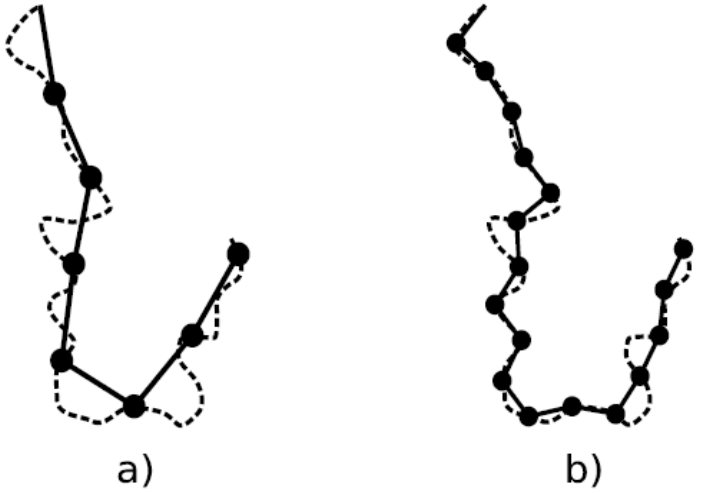

Figura 8. La escala de la unidad de la medida empleada modifica el resultado obtenido.

El primer grupo de pruebas se realizó con un conjunto de municipios de Cuba representados en una escala 1:500 000; comparándose los resultados arrojados por el algoritmo propuesto y los resultados obtenidos para los mismos polígonos, utilizando MapInfo y las función ST_Area de Postgis transformando las coordenadas a las proyecciones EPSG:2085 y EPSG:2086 respectivamente. Los resultados de estas mediciones se muestran en la Tabla 2 y Figura 9 expresados en kilómetros cuadrados respectivamente.

Tabla 2. Valores de las mediciones de áreas de polígonos en escala 1:500 000.

Región	MapInfo	Postgis (2085)	Postgis (2086)	Algoritmo Propuesto
San Juan y Martínez	405,009	403,9143283	404,2283507	404,2779931
San Antonio de los Baños	130,43	130,1017484	130,2797945	130,2089398
Madruga	447,984	446,869987	447,510497	446,9620815
Los Arabos	748,965	747,0036781	747,8447752	749,5081552
Jovellanos	504,829	503,5480719	504,2183746	507,3098716

Colón	595,895	594,3571673	595,0894715	597,5460233
Cruces	196,655	196,1230512	196,2762825	198,6428112
Placetas	603,122	601,4902318	601,9473072	602,4329133
Manicaragua	1.062,70	1059,8104	1060,479165	1063,362983
Santo Domingo	875,7	873,4060305	874,3914672	875,3329034
Ranchuelo	549,672	548,1981547	548,6906695	551,6447314
Quemado de Güines	316,486	315,6772997	316,0841249	318,3506661
Fomento	465,461	464,1972602	464,4477088	466,0971347
La Sierpe	1.006,74	1004,059181	1004,222441	1009,323201
Sagua de Tánamo	706,746	709,5722342	705,3414425	704,6596852
Frank País	499,753	498,7042331	498,2829088	497,8296829

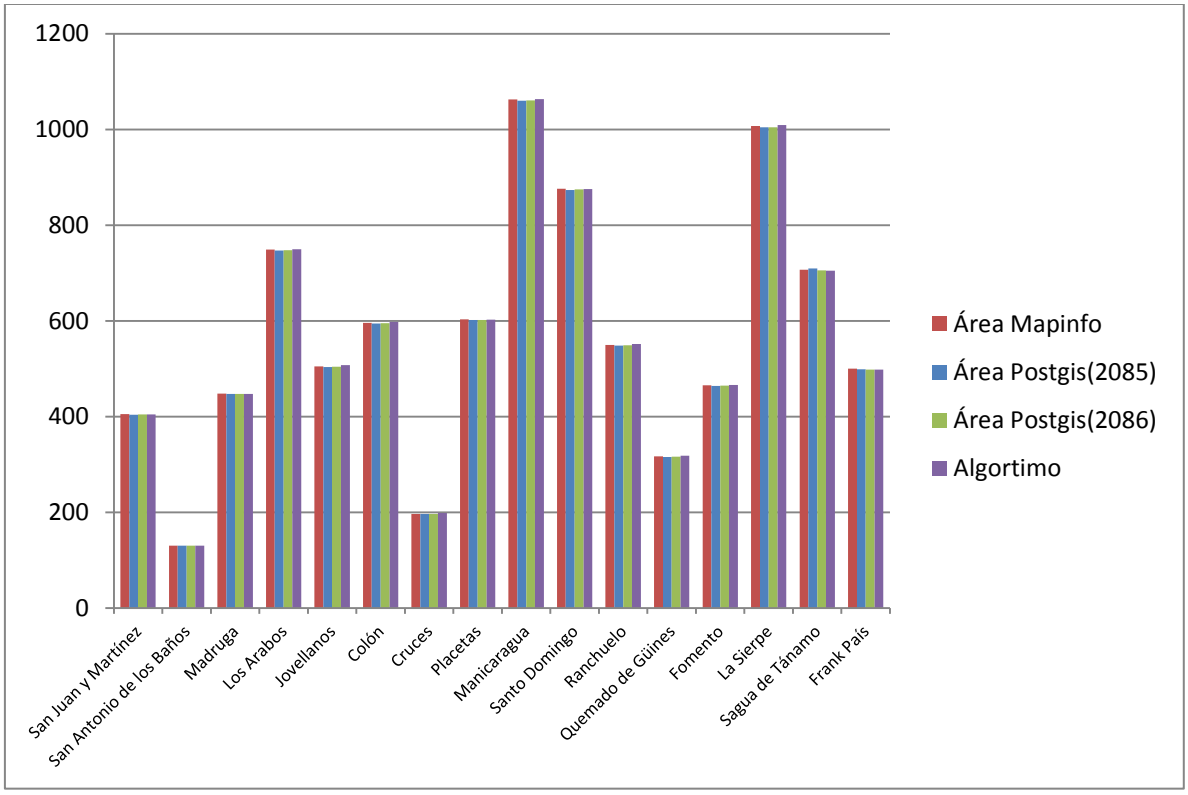

Figura 9. Comparación gráfica de los resultados obtenidos para la escala 1:500 000.

Para el segundo grupo de prueba se utilizaron los mismos municipios, pero en esta ocasión en una representación en escala 1:250 000. Los resultados de estas mediciones se muestran en la Tabla 3 y Figura 10 expresados en kilómetros cuadrados respectivamente.

Tabla 3. Valores de las mediciones de áreas de polígonos en escala 1:250 000.

Región	MapInfo	Postgis (2085)	Postgis (2086)	Algoritmo Propuesto
San Juan y Martínez	406,645	405,5461961	405,861339	400,8808058
San Antonio de los Baños	130,265	129,936429	130,1142107	135,5689788
Madruga	447,808	446,6944101	447,3346966	442,8680584
Los Arabos	747,423	747,0036781	747,8447752	753,5555376
Jovellanos	505,313	504,0306926	504,7013864	503,6723832
Colón	597,879	596,3366151	597,0712043	602,6732356
Cruces	196,411	195,8798178	196,0328505	196,7046385
Placetas	602,52	600,8902879	601,3469224	601,0251069
Manicaragua	1.063,43	1060,540882	1061,210065	1057,31177
Santo Domingo	875,458	873,4060305	874,3914672	877,7535266
Ranchuelo	549,981	548,3702718	548,505862	548,3702718
Quemado de Güines	315,899	315,0914413	315,4974845	310,8600063
Fomento	464,363	463,1020192	463,3519641	459,1251257
La Sierpe	1.005,29	1002,6077	1002,770431	1010,147319
Sagua de Tánamo	707,502	706,0961694	705,4135695	706,6139609
Frank País	501,984	500,930477	500,507247	502,2789209

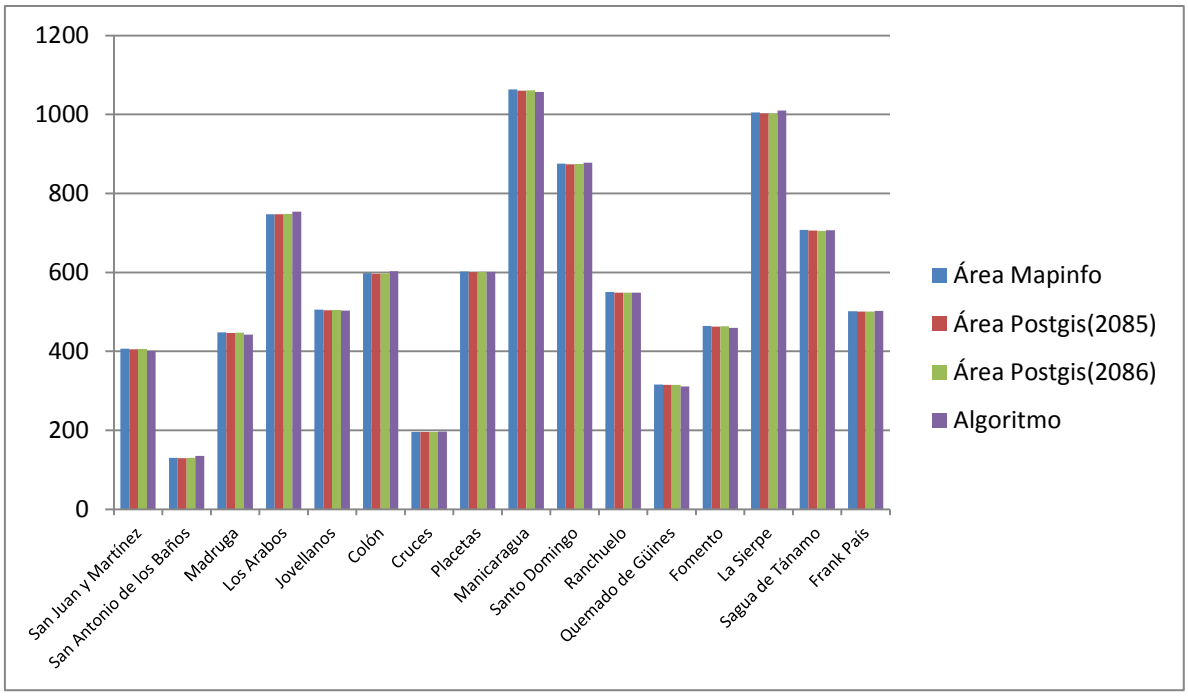

Figura 10. Comparación gráfica de los resultados obtenidos para la escala 1:250 000

Y para un tercer y último grupo de prueba se utilizó otro conjunto diferente de polígonos –Capitales Provinciales- en una escala menor que 1:100 000. Los resultados en la Tabla 4 y Figura 11.

Tabla 4. Valores de las mediciones de áreas de polígonos en escala menor que 1:100 000.

Región	MapInfo	Postgis (2085)	Postgis (2086)	Algoritmo Propuesto
Bayamo	10,0381	10,01960538	10,00846604	10,98275112
Las Tunas	16,5926	16,55381367	16,54491293	14,12287383
Camagüey	63,1462	62,983798	62,97622072	68,68977622
Ciego de Ávila	12,9559	12,92088205	12,92511921	13,16746572
Pinar del Río	13,3728	13,33696663	13,34908336	10,06496844
Cienfuegos	18,8189	18,76778513	18,77975989	18,47725493
Sancti Spíritus	11,6092	11,57767856	11,58256003	16,10096355
Santiago de Cuba	40,1734	40,11536241	40,05706149	44,62707222

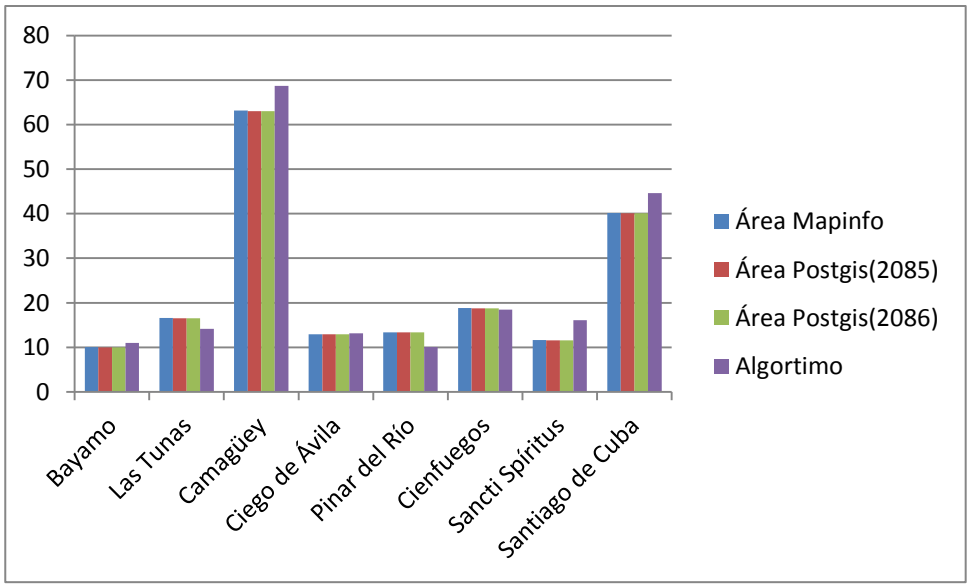

Figura 11. Comparación gráfica de los resultados obtenidos para la escala menor que 1:100 000.

3.3. Conclusiones Parciales

Debe tenerse en cuenta que en el proceso para el cálculo del área de polígonos en coordenadas geográficas o elipsoidales con el algoritmo propuesto se introducen una serie de errores de diferentes naturalezas. Como es el caso de los inherentes al modelo esferoide/elipsoide empleado respecto al modelo real de la tierra, los introducidos por el modelo esférico utilizado en la concepción del algoritmo y los introducidos por el cálculo en un ordenador. Se puede observar que los resultados son completamente satisfactorios pues son muy similares a los obtenidos por otras herramientas tanto privativas como libres. De esta forma se demuestra que la propuesta permite obtener resultados válidos de forma automática a usuarios poco experto en temas de cartografía y geodesia.

Por otro lado las pruebas arrojaron que el factor escala no afecta completamente los resultados arrojados por el algoritmo propuesto, aunque si se analiza el error relativo de los resultados obtenidos por el algoritmo propuesto respecto a las demás herramientas se puede apreciar que este aumenta notablemente para escalas menores que 1:100000 principalmente.

CONCLUSIONES

- El análisis geométrico en los SIG no es un problema simple si se quieren obtener resultados válidos. Debe el usuario ser consciente de la naturaleza de los datos con los que está trabajando, qué tipo de proyección y algoritmos utilizar en dependencia de los resultados que se quieran obtener.

- La aproximación del esferoide/elipsoide a una esfera permite obtener algoritmos sencillos y válidos, con un mínimo de sacrificio en la exactitud de los resultados, para escalas mayores que 1:100000, para cálculos geométricos sobre su superficie.

- Se elimina la utilización de algoritmos geodésicos cuando se trabaja en coordenadas geográficas, obteniéndose algoritmos eficientes de complejidad lineal sin la necesidad de imposición de restricciones en la digitalización de las geometrías.

- La combinación de estos algoritmos con los clásicos euclidianos, teniendo en cuenta las características de los datos, permite darles a usuarios avanzados la opción de elegir con qué tipo de método desea trabajar. Y a usuarios inexpertos en temas de cartografía y geodesia les garantiza obtener resultados válidos independientes del sistema de coordenadas con el que trabaje.

RECOMENDACIONES

- Probar la propuesta con un mayor número de polígonos relativamente pequeño para un mejor análisis el comportamiento del algoritmo propuesto para el área de polígonos. Dígase, polígonos contenido en una hoja cartográfica, con un área menor a 25 kilómetros cuadrados.

- Incluir en la propuesta para el cálculo de la distancia tomar en cuenta las dimensiones de los semiejes del elipsoide de referencia.

- Incluir la propuesta en los SIG y/o componentes de este tipo, libres disponibles actualmente.

REFERENCIAS BIBLIOGRÁFICAS

Alonso Sarría, Francisco. 2006. SIG y Teledetección en la Universidad de Murcia. *www.um.es.* [En línea] Junio de 2006. http://www.um.es/geograf/sigmur/sigpdf/temario.pdf.

Barrero Ripoll, Manuel, y otros. 2008. *Trigonometía Esférica Fundamentos.* Madrid : E.T.S.I en Topografía, Geodesia y Cartografía, 2008. ISBN:84-96244-13-x.

Bevis, Michael y Cambareri, Greg. 1987. Computing the Area of a Spherical Polygon of Arbitrary Shape. *Mathematical Geology.* 1987, Vol. 19, 4.

Brent, Richard P. 1976. Fast Multiple-Precision Evaluation of Elementary Functions. *Journal of the Associataon for Computang Machinery.* 1976, Vol. 23, 2.

Chen, J. 1996. *Computational Geometry - Methods and Appilcations.* Texas : A&M University, Computer Science Departament, 1996.

Coelho de Pina, José. 2004. www.ime.usp.br. *Instituto de Matemática e Estadística.* [En línea] 2004. http://www.ime.usp.br/~cris/mac331/intro.pdf.

Evenden, Gerald I. 2003. *Cartographic Projection Procedures for the UNIX.* s.l. : UNITED STATES DEPARTMENT OF THE INTERIOR. Geological Survey, 2003.

Fallas, Jorge. 2003. *PROYECCIONES CARTOGRÁFICAS Y DATUM ¿Qué son y para qué sirven?* TeleSig-Universidad Nacional : Laboratorio de Teledetección y Sistemas de Información Geográfica PRMVS-EDECA, 2003.

García Alvarado, Martín Giraldo. 2002. El Siglo de la Geometria. *Apuntes de Historia de las Matématicas.* mayo de 2002, Vol. 1, No. 2.

Goizueta, Javier. 2006. *Sistemas de Referencia Geodésicos. Proyecciones Cartográficas.* s.l. : ECAS Técnicos Asociados S.A., 2006.

Iglesias, M. y M., A. 1996. *Trigonometría Esférica. Breve Introducción a la navegación.* Universidad del País Vasco : Servicio Editorial, 1996.

Iskategia, Kepler. 2001. *Trigonometría Esférica.* 2001.

Mencia, J. 2006. *Nociones de trigonometría esférica.* Universidad del País Vasco-Euskal Herriko. : s.n., 2006.

MTL. 2008. Calculate distance, bearing and more between Latitude/Longitude points. *Movable Type Scripts.* [En línea] MTL, 2008. [Citado el: 19 de febrero de 2011.] http://www.movable-type.co.uk/scripts/latlong.html.

Olaya, Victor, y otros. 2007. *Sistemas de Información Geográfica.* s.l. : OSGEO, 2007.

Ortiz, Gabriel. 2003. gabrielortiz.com. *Tu web sobre Sistemas de Información Geográfica GIS- SIG.* [En línea] 2003. http://recursos.gabrielortiz.com/index.asp?destino=diccionario&termino=geoide%20%28geoid%29.

Peña Lloips, Juan. 2006. *Sistemas de Informacion Geográfica Aplicados a la Gestión del Territorio.* Universidad de Alicante : Editorial Club Universitario , 2006. ISBN:84-8454-493-1.

Pesquer Mayos, Lluís, Pons Fernández, Xavier y Masó Pau, Joan. 2003. International Society for Photogrametry and Remote Sensing. *www.isprs.org.* [En línea] septiembre de 2003. http://www.isprs.org/publications/related/semana_geomatica05/front/abstracts/Dimarts8/G08_abs.pdf.

Refractions Research, Inc. 2008. *Postgis Manual 1.5.0.* Victoria, British Columbia, Canada : Refractions Research Inc, 2008.

S. Chinea, Carlos. 2002. *LAS FÓRMULAS DE LA TRIGONOMETRIA ESFERICA.* 2002.

W. Weisstein, Eric. 2011. Bijective. *WolframMathWorld.* [En línea] Wolfram Research, Inc., 2011. http://mathworld.wolfram.com/Bijective.html.

Zakatov, P.S. 1981. *Curso de Geodesia Superior.* s.l. : Editorial Mir, 1981.

BIBLIOGRAFIA CONSULTADA

Alonso Weber, Juan Manuel. 2008. *Un Nuevo Modelo Autoorganizado Aplicado a la Resolución de Problemas de Geometría Computacional.* 2008. Tesis Doctoral.

Álvarez Alonso, Marina. 2007. *Modelos de datos en un SIG.* Madrid, España : Universidad Politécnica de Madrid, Facultad de Informática, 2007.

Boland, Ralph P. y Urrutia, Jorge. 2001. *Polygon Area Problems.* [ed.] McGuill Universit. Canada : In Proceding of the12th Canadian Conference on Computational Geometry, 2001.

Contreras-Alcalá, Felipe. 1998. *Cutting polygons and a problems on illumination stages.* Dept. Compt. Ottawa Ont, Canada : Sci. University of Ottawa, 1998. Masters Thesis.

Czyzowics, Jurek, Contreras-Alcalá, Felipe y Urritia, Jorge. 1998. *On measuring area of polygons.* [ed.] McGuill University. Canada : In Proceedings of the 10th Canadian Conference on Computacional Geometry, 1998.

de Berg, M. y Van Kreveld, M. 1997. *Computational Geometry - Algorithms and Applications.* s.l. : Springer, 1997.

Farell, J. y Barth, M. 1998. *El Sistema de Posicionamiento Global y la Navegación Inercial.* EE.UU : s.n., 1998.

Gallardo, D. 1999. Sección 6, Triangulación de Polígonos. Universidad de Alicante. : DCCIA, 1999.

Galo, Mauricio, Galera Monico, João F. y Castro de Oliveira, Leonardo. *CÁLCULO DE ÁREAS DE POLÍGONOS SOBRE O ELIPSÓIDE USANDO PROJEÇÕES EQUIVALENTES.* s.l. : Anais do III Colóquio Brasileiro de Ciências Geodésicas.

Gudmundsson, Sigmundur. 1996. *An Introduction to Riemannian Geometry.* s.l. : Lund University, 1996.

H. Cormen, T., E. Leirserson, C. y L. Rivest, R. *Introduction to Algorithms.*

Jackson, J.E. 1980. *Sphere, Spheroid and Projection.* London : Granada, 1980.

Lafuente López, Javier. 2002. *GEOMETRIA DIFERENCIAL DE CURVAS.* 2002.

Lauf, G. B. 1983. *Geodesy and Map Projection.* Victoria, Australia : TAFE publications, 1983.

Martín Asín, F. 1983. *Geodesia y Cratografía Matemática.* Madrid : Paraninfo S.A., 1983.

Open Geospatial Consortium Inc. 2010. *OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option.* s.l. : Open Geospatial Consortium Inc., 2010. OGC 06-104r4.

P. Preparata, F. y Shamos, M. I. *Computational Geometry an Introduction.*

Ramón Antunez, Romanuel y Hernández Montero, Lidisy. 2010. *Algoritmo para el cálculo del área de polígonos curvos.* La Habana, Cuba : JCJ-ICIMAF, 2010.

Ramón Antunez, Romanuel y Hernández Montero, Lidisy. 2010. *Algoritmo para el cálculo del área de polígonos sobre la tierra.* Mérida, Venezuela : COVESIG, 2010.

Ramón Antunez, Romanuel y Hernández Montero, Lidisy. 2011. *Modelo de utilización de una propuesta de algoritmos para análisis geométricos en SIG.* Morelia Michoacán, México : Reunión Nacional SELPER-MEXICO, 2011.

Richardus, P. y Adler, R.K. 1972. *Map Projection from geodist, cartogrphers and geographers.* Amsterdam : North-Holland, 1972.

Robinson, A.H., y otros. 1984. *Elements of Cartography.* New york : John Wiley and Sons, 1984.

Rodriguez Roche, DrC. Ernesto. 2009. *El problema de la transformación de coordenadas determinadas con GPS.* La Habana, Cuba : Departamento de Geodesia y Cartografía. GEOCUBA Investigacion y Consultoría, 2009.

SEDGEWICK, ROBERT. 1983. GEOMETRIC ALGORITHMS. *ALGORITHMS.* Menlo Park, California : Addison-Wesley Publishing Company, Inc., 1983.

Snyder, J.P. 1987. *Map Projection - A Working Manual.* Washinton : United State Goverment Printing Office, 1987.

Valenzuela Ruz, Víctor. INACAP. *INACAP.* [En línea] www.. informatica.inacap.cl.