Escola Politècnica Superior de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE DE FI DE CARRERA

TÍTOL DEL PFC : Desenvolupament d'una eina de planificació de vol per UAVs

TITULACIÓ: Enginyeria Tècnica Aeronàutica, especialitat Aeronavegació

AUTOR: Marcos Pérez Batlle

DIRECTOR: Eduard Santamaria Barnadàs

DATA: 30 de setembre de 2008

Títol : Desenvolupament d'una eina de planificació de vol per UAVs

Autor: Marcos Pérez Batlle

Director: Eduard Santamaria Barnadàs

Data: 30 de setembre de 2008

Resum

Aquest document té com a finalitat el desenvolupament d'una eina de planificació de vol per Aeronaus no Tripualdes (UAV). Per fer-ho, primerament es realitza una introducció general al programa ideat i una explicació dels seus modes de funcionament. Posteriorment es descriu la teoria i la implementació de cada bloc que forma el programa per separat. En el capítol següent es parla dels softwares utilitzats i es descriu el codi breument, classe a classe. Per acabar es proposen tres casos pràctics sobre el funcionament dels diferents modes de l'eina. És en aquesta part del treball on es posa en evidència el funcionament en conjunt de tots els diferents blocs que formen el programa.

L'eina de planificació de vol per UAVs té com a objectiu principal crear les bases per realitzar un pla de vol de manera automàtica a partir d'unes dades reduïdes d'entrada, que depenen del mode de funcionament de la mateixa eina. No obstant, els tres modes es basen en el mateix procediment de funcionament: A partir d'uns obstacles i una altura de vol definida per l'usuari o per la pròpia eina es crea un diagrama de rutes segures al voltant dels obstacles de tal manera que, seguint les arestes d'aquest diagrama es poden unir dos punts sense colisionar amb cap obstacle. Posteriorment, es cerca la ruta òptima entre aquests dos punts i es representen els resultats obtinguts mitjançant Google Earth. Aquesta eina ha estat implementada mitjançant el llenguatge Csharp, de la plataforma .NET.

Title : Flight planning development tool for UAVs

Author: Marcos Pérez Batlle

Director: Eduard Santamaria Barnadàs

Date: September 30, 2008

Overview

This document has the development of a tool of flight planning as a purpose for Unmanned Air Vehicles (UAV). To make it, firstly a general introduction to the devised program and an explanation in its ways of operation are carried out. Later the theory and the implementation of each block that forms the program separately are described. The following chapter is about the used software and the code portrays, class by class, briefly. For ending, three practical cases about the operation of the different modes of the tool are proposed. It is in this part of the work where the operation of all the different blocks that form the program is shown.

The main goal of this flight planning tool for UAVs is to create the bases for carrying out a flight plan in automatic way from a reduced entry datum, which depend in the way of operation of the same tool. However, the three modes have the same operation procedure: From a set of obstacles and a altitude of flight defined by the user or by the tool a diagram of safe routes is created among the obstacles in such way that, following the edges of this diagram two points can be joined without crashing with any obstacle. Later, the optimum route is looked between these two points and the results obtained through Google Earth are represented.

This tool has been implemented through the language Csharp, of the platform .NET

com anar al cel i tornar, per la Maria

ÍNDEX

ÍNDEX DE FIGURES

INTRODUCCIÓ

La innovació tecnològica en l'àmbit civil i industrial té com a objectiu la millora qualitativa de la seguretat i l'eficiència. Si ens centrem en la indústria aeronàutica aquests dos objectius estan a l'ordre del dia. El transport aeri és considerat el mitjà de transport més segur i l'alta competència entre companyies i operadors aeris asseguren una eficiència considerable en el mercat per la pròpia subsistència de les mateixes. En aquest sentit els UAV[1] han estat ideats per aquelles operacions on l'única manera d'assolir els objectius de seguretat i eficiència és supervisant o controlant l'aeronau des d'un indret remot sense la necessitat d'un pilot a bord.

Un exemple d'operació civil on els UAVs podrien jugar un paper rellevant és en tasques de prevenció d'incendis. Poder tenir una flota de UAVs equipada amb els instruments necessaris per analitzar una regió i determinar les zones amb més risc d'incendi és més segur a l'hora d'evitar riscos innecessaris als pilots, i més eficient si considerem la flota de UAVs de baix cost. Altres operacions civils on els UAVs podrien destacar podrien ser en el servei aeronàutic de cerca i rescat (SAR) de qualsevol estat.

Un altre punt important a tenir en compte és la gran diferència entre els UAVs i les aeronaus que funcionen per control remot. Encara que sembli que parlem del mateix tipus d'aeronaus, un UAV és capaç de volar sense la necessitat de que ningú el controli des de terra, això si, a partir d'unes ordres o condicions preestablertes, és a dir, pot maniobrar lliurement dins d'un marc preestablert i, per tant, pot realitzar tasques que les aeronaus convencionals no poden sense que la vida de cap pilot corri cap mena de risc.

Com es pot apreciar, els UAVs poden tenir infinitat d'aplicacions gràcies a la seva versatilitat, però suprimir la figura del pilot porta a la necessitat d'un pilot automàtic que controli cada moviment de l'aeronau, uns sensors capaços de detectar variacions de qualsevol moviment, i el que no és menys important: el marc preestablert anomenat abans, una planificació acurada del vol a dur a terme.

Figura 1: Un UAV civil, el Watchkeeper (Israel)

L'objectiu d'aquest projecte és l'elaboració d'una eina de planificació de vol per UAVs. Aquesta eina tindria la capacitat de, a partir de l'entrada d'uns punts per on UAV no pot

[1] Unmanaged Air Vehicle: Aeronau no tripulada

sobrevolar (obstacles de qualsevol mena, zones poblades, etc.) obtenir la ruta més eficient entre dos punts passant el més lluny possible dels propis obstacles. És aquí on entren en joc les diferents parts d'aquest projecte.

Necessitem, primerament, un diagrama capaç d'indicar-nos totes les possibles rutes entre el punt de partida de l'aeronau i el punt d'arribada que passin el més lluny possible dels obstacles però que siguin eficients, és a dir, no ens serveix una ruta que rodegi els obstacles si hi ha una altra que els travessa si segueix essent segura. De tots els possibles diagrames existeix un que s'adequa perfectament a les especificacions: el diagrama de Voronoi, que s'explica detalladament en el segon capítol d'aquesta memòria.

A més a més necessitem un algorisme capaç de trobar quina és la ruta més eficient però que no posi en compromís la seguretat de l'aeronau. Aquest ha de ser capaç de ponderar distància total, consum de combustible, angles de gir o qualsevol restricció que l'usuari final trobi pertinent. S'ha implementat un algorisme que satisfà aquestes necessitats.

El software resultant té tres modes de funcionament ben diferenciats que s'expliquen acuradament al llarg d'aquesta memòria. No obstant aquets modes tenen bases similars. En poques paraules, s'ha dissenyat un programa que és capaç de trobar una ruta entre dos punts donats **a altura de vol constant** òptima en distància i plausible per un UAV a partir de models digitals d'elevacions del terreny o de la introducció manual d'obstacles. Es trata d'una eina que funciona off-line, és a dir, el seus resultats poden ser utilitzats pel disseny d'un pla de vol. Com s'ha comentat abans s'encarega de fer realitat el marc preestablert, necessari pel bon funcionament d'un UAV.

La implementació de tota l'eina de planificació de vol s'ha realitzat mitjançant el software Microsoft Visual Csharp 2005 i utilitza, per presentar gràficament els resultats obtinguts el software Google Earth. Aquestes dues eines seran profunditzades en aquesta mateixa memòria més endavant.

CAPÍTOL 1. DIAGRAMA GENERAL DEL PROGRAMA

1.1. Introducció

En aquest capítol es donarà una visió general sobre l'estructura de l'eina de planificació de vol en els seus tres modes de funcionament. Per a cada un d'aquests es detallaran les dades que l'eina necessita de l'usuari per funcionar correctament, les diferents parts o blocs en que està estructurat cada mode i els resultats finals que genera.

Cada bloc dels diagrames que es presenten a continuació està implementat com un objecte diferent en el programa per afavorir la modularitat, la comprensió i la possible modificació per part de l'usuari del codi de la pròpia eina.

Encara que els tres modes siguin diferents entre si, tots ells es basen en els mateixos principis: a partir d'unes dades introduïdes per part de l'usuari es generen diagrames de Voronoi (del qual es parla més acuradament en el capítol següent) per esquivar els possibles obstacles que puguin interposar-se pel camí, s'utilitza l'algorisme de la recta (del qual també es parlarà amb més detall després) en el cas de que no hi hagi obstacles i un cop es tenen tots els possibles camins es cerca el més òptim ponderant diversos factors a tenir en compte (distància, angles de gir, etc.). Un cop es troba el resultat, aquest es presenta mitjançant Google Earth.

1.2. Primer mode

1.2.1. Les dades d'entrada

En aquest primer mode, l'usuari ha d'introduir manualment les coordenades dels obstacles mitjançant Google Earth[1]. A més es necessita el model digital d'elevacions del terreny a sobrevolar, del qual se'n parla en el capítol següent, i l'atura de vol de l'aeronau.

1.2.2. Estructuració del programa

En aquest mode, el programa estructurat estructurat com si indica a la figura1.1.

Un cop el programa obté per part de l'usuari les dades d'entrada, genera el diagrama de Voronoi a partir del conjunt d'obstacles, i cerca el camí òptim entre dos nodes indicats per part de l'usuari.

[1] En el capítol 4 es realitza un exemple sobre com introduir obstacles.

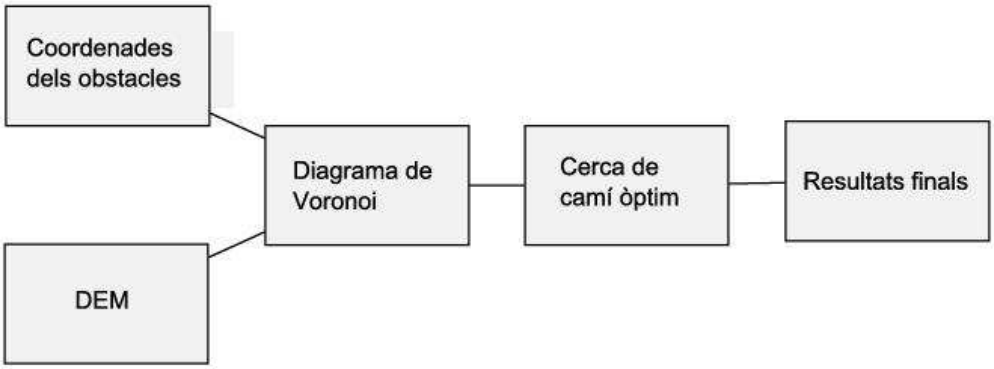

Figura 1.1: Diagrama de blocs de l'eina de planificació de vol operant en el primer mode.

1.2.3. Els resultats finals

Com a resultat final es presenten diferents arxius en un format específic anomenat KML, del qual es parla amb més detall en el tercer capítol. Aquest format és compatible amb Google Earth. Un exemple de resultat final és el que es presenta en la figura 1.2.

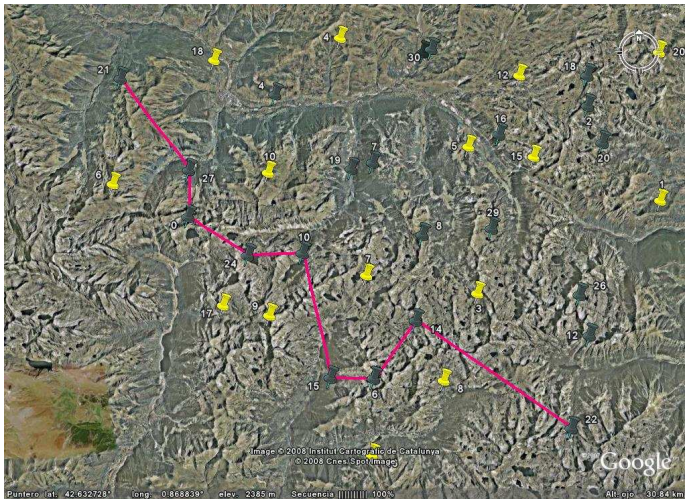

Figura 1.2: Exemple de resultat final en el mode de funcionament 1 del programa

1.3. Segon mode

1.3.1. Les dades d'entrada

En aquest ja no cal que l'usuari introdueixi manualment els obstacles sinó que serà el propi programa que els generarà. En aquest cas, l'usuari només introduirà dos parells de coordenades que marcaran el marc territorial on es vol calcular el camí òptim, l'alçada de vol de l'aeronau i el model d'elevacions digital.

1.3.2. Estructuració del programa

En aquest mode, el programa està estructurat tal i com s'inidica a la figura 1.3.

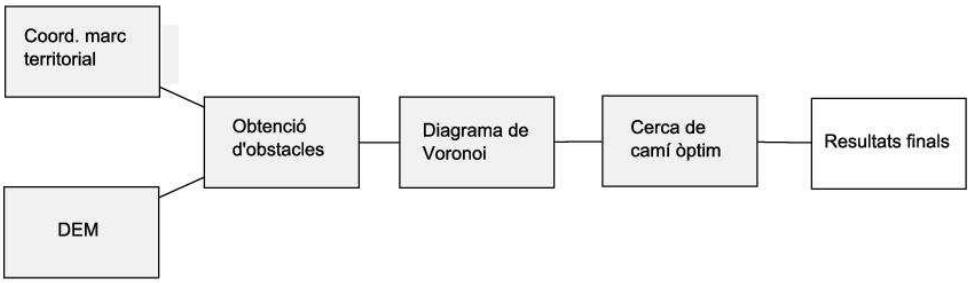

Figura 1.3: Diagrama de blocs de l'eina de planificació de vols operant en el segon mode.

En aquest mode, un cop s'han carregat les dades inicials al programa, l'obtenció d'obstacles s'efectua trobant, primerament, les corbes de nivell a una cota igual a l'altura de vol de l'aeronau. D'això s'encarrega un conjunt de llibreries anomenades GDAL (Geospatial Data Abstraction Library) de les quals se'n parla més acuradament en el següent capítol. Un cop s'obtenen els obstacles, tot funciona de la mateixa manera que en el primer mode; es generen el diagrama de vol pertinent i es cerca el camí òptim entre dos nodes indicats per l'usuari.

1.3.3. Els resultats finals

Els resultats finals no difereixen del mode de funcionament anterior; es creen arxius en format KML per a la seva posterior representació en Google Earth. Un exemple de resultat final en aquest mode de funcionament és el de la figura 1.4.

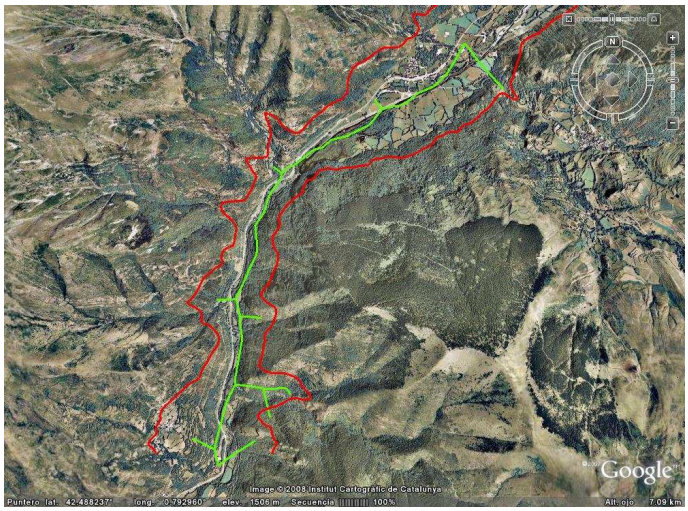

Figura 1.4: Exemple de resultat final del segon mnode de funcionament del programa

1.4. Tercer mode

1.4.1. Les dades d'entrada

En aquest mode de funcionament, l'usuari ja no cal que introdueixi el marc territorial del que es parlava en el mode anterior. L'usuari, a part d'introduir el model digital d'elevacions i l'altura de vol, només ha d'entrar les coordenades dels punts d'origen i destinació de l'aeronau.

1.4.2. Estructuració del programa

En aquest mode de funcionament, el programa està estructurat tal i com s'inidica en la figura 1.5:

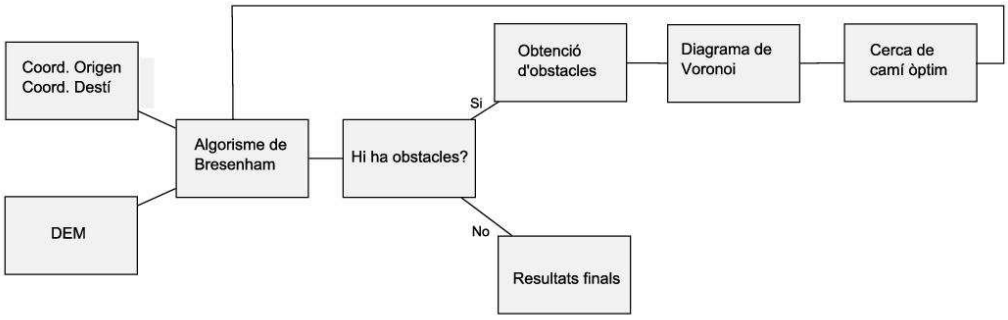

Figura 1.5: Diagrama de blocs de l'eina de planificacio de vol

Aquest és el mode de funcionament més complexe de tots: Com que el camí més curt entre dos punts és la recta, l'algorisme de l'equació de la recta que del qual ja en parlarem més endavant, per determinar l'existència d'obstacles en la trajectoria de l'aeronau. Si no hi ha cap obstacle, és evident que la ruta més eficient en quant a distància i seguretat serà la recta. En cas contrari, si algun obstacle s'interposa entre el punt d'origen i el punt de destí es crea automàticament un marc territorial com el del segon mode al voltant de l'obstacle per generar un diagrama de Voronoi per esquivar-lo i es busca el camí òptim dins d'aquest diagrama. Un cop s'ha salvat l'obstacle es torna a utilitzar l'algorisme de la recta per arrivar al punt de destí. Si ens tornem a trobar obstacles es torna a crear el marc territorial, en cas contrari, s'ha arribat a la destinació.

1.4.3. Els resultats finals

Els resultats finals no difereixen gaire dels modes de funcionament anteriors. La única diferència és la major generació d'arxius KML al implicar dos algorismes totalment diferents (realització de diagrames de Voronoi i l'algorisme de la recta) per arrivar a l'objectiu. Un exemple de resultat final en aquest mode de funcionament és el de la figura 1.6.

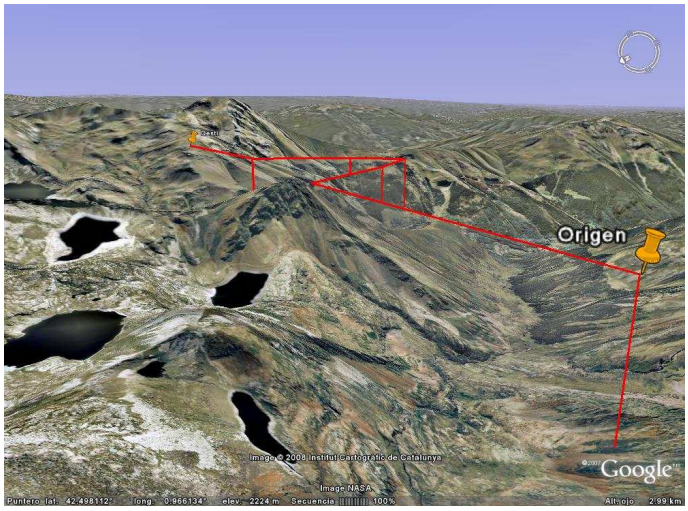

Figura 1.6: Exemple de resultat final del tercer mode de funcionament del programa

CAPÍTOL 2. L'EINA DE PLANIFICACIÓ DE VOL BLOC A BLOC. TEORIA I IMPLEMENTACIÓ

2.1. Introducció

En aquest capítol es donarà una visió teòrica general sobre cada bloc que conforma l'eina de planificació de vol per UAVs a més dels principals trets carecterístics de la implementació seguida al programa. No es tracta d'una explicació exhaustiva del codi.

A més, es dediquen les dos últimes seccions a detalls de funcionament molt importants en l'eina com són la viabilitat dels viratges i l'obtenció de rutes vàlides a partir d'obstacles no puntuals.

2.2. Sistemes de refèrencia

2.2.1. Introducció

Si bé els sistemes de referència no formen en si materixos un bloc en el programa, se li ha dedicat una secció com si ho fossin, ja que al llarg de tota la implementació del programa s'ha de tenir en compte amb quin sistema de referència s'està treballant en cada moment, com es veurà més endavant, a les explicacions de cada bloc. En aquest projecte es treballa amb coordenades geogràfiques i coordenades UTM[1] referides a dos datums diferents com es veurà a continuació. Per tant, és important definir els conceptes següents.

2.2.2. Les coordenades geogràfiques

El sistema de coordenades geogràfiques expressa totes les posicions de la Terra en dues de les tres coordenades d'un sistema de coordenades esfèric que està alineat amb l'eix de rotació de la Terra. Aquest defineix dos angles mesurats des del centre de la Terra; la latitud (mesura l'angle de qualsevol punt de la Terra i l'equador de la mateixa) i la longitud (mesura l'angle seguint l'equador de qualsevol punt de la Terra i el meridià de Greenwich). Combinant aquest dos angles es pot expressar la posició de qualsevol punt de la superfície de la Terra com es pot comprovar en la següent figura 2.1 [1].

[1] UTM: Universal Transversal Mercator

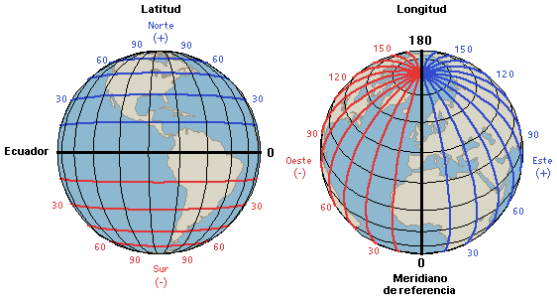

Figura 2.1: Coordenades geogràfiques.

2.2.3. Sistemes de coordenades UTM

El sistema de coordenades Universal Transversal Mercator és un sistema basat en la projecció geogràfica cilíndrica tangent a dos meridians. En altres paraules, es tracta d'envoltar la Terra en un cilindre tangent a un meridià i al seu respectiu antimeridià, projectar tots els punts de la Terra sobre el cilindre, retallar-lo longitudinalment i obrir-lo. A diferència del sistema de coordenades anterior on qualsevol punt estava expressat amb dos angles (latitud i longitud), les magnituds del sistema de coordenades UTM estan expressades en metres a nivell del mar, que es la base de la projecció de l'el·lipsoide de referència, del qual es parla en la següent secció [2].

Les figures 2.2 i 2.3 grafiquen el que s'ha comentat en el paràgraf anterior.

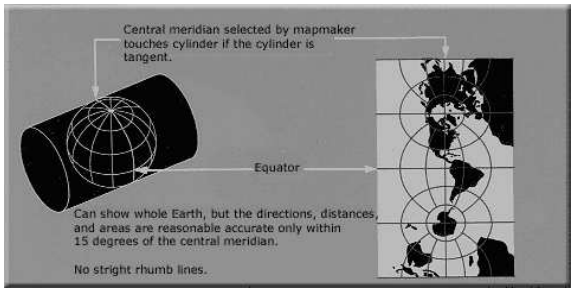

Figura 2.2: Projecció de Mercator transversa.

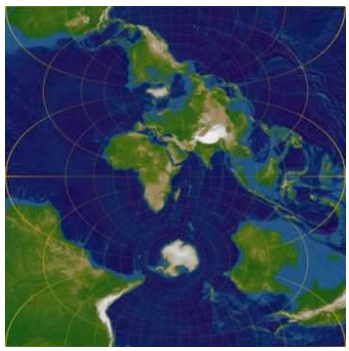

Figura 2.3: Mapa del món en projecció transversa Mercator, centrat en el meridià 45 E i l'Equador.

2.2.4. Els el·lipsoides de referència

Els el·lipsoides de referència són aproximacions de la forma de la Terra en el·lipsoides sense considerar les ondulacions pròpies de la topografia. Això es deu a que és una figura matemàtica fàcil de manipular i suficientment semblant a la forma de la Terra. Ara bé, al llarg de finals del segle XIX i durant tot el segle XX s'han anat donant diferents valors als paràmetres més importants de l'el·lipsoide per aproximar-los d'una manera més acurada a la forma real de la Terra [3].

A la taula 2.1 es donen els valors dels el·lipsoides més populars en geodèsia.

Nom	**Semieix major (m)**	**semieix menor (m)**	$1/f$
Australian National	6378160.000	6356774.719	298.250000
Bessel 1841	6377397.155	6356078.963	299.152813
Clarke 1866	6378206.400	6356583.800	294.978698
Clarke 1880	6378249.145	6356514.870	293.465000
Everest 1956	6377301.243	6356100.228	300.801700
Fischer 1968	6378150.000	6356768.337	298.300000
GRS 1980	6378137.000	6356752.314	298.257222
International 1924 (Hayford)	6378388.000	6356911.946	297.000000
SGS 85	6378136.000	6356751.302	298.257000
South American 1969	6378160.000	6356774.719	298.250000
WGS 72	6378135.000	6356750.520	298.260000
WGS 84	6378137.000	6356752.314	298.257224

Taula 2.1: Paràmetres del el.lipsoides de referència

2.2.5. Implementació

A l'eina de planificació de vol es treballa fonamentalment amb els dos sistemes de coordenades anteriors, els sistema de coordenades geogràfiques i el sistema de coordenades UTM referits a dos el·lipsoides de refèrencia diferents. Les coordenades geogràfiques estan referides al el·lipsoide WGS84, mentre que les coordenades UTM estan referides al datum ED50[2], que utilitza com a el·lipsoide de referència el Heyford International 1924. Això es degut a que Google Earth treballa amb coordenades geogràfiques referides al WGS84 a més de les funcions que computen els diagrames de Voronoi, mentre que els mapes DEM treballen amb projeccions UTM referides al Heyford International 1924. Aquest fet obliga a realitzar constants transformacions de coordenades entre el·lipsoides i de geogràfiques a projectades.

En el programa hi ha un objecte encarregat de realitzar totes aquestes transformacions basades en les fórmules de Coticchia-Surace [4] en les quals no es profunditza en aquesta memòria.

[2]ED50: European Datum 1950, sistema de referència oficial a Europa.

2.3. Els mapes DEM

2.3.1. Introducció. Els models digitals del terreny

Es denomina model digital del terreny a una estructura numèrica de dades que representa la distribució espaial d'una variable quantitativa i contínua, com pot ser la temperatura, la cota o la pressió atmosfèrica, per exemple. En particular, quan la variable a representar és la cota o altura del terreny se'l denomina Model Digital d'Elevacions o MDE (DEM en anglès).

A les figures 2.4 i 2.5 es mostra un exemple d'aplicació dels models digitals del terreny [5].

Figura 2.4: Paissatge de la Selva Negra (Baden-Wurtemberg, Alemanya).

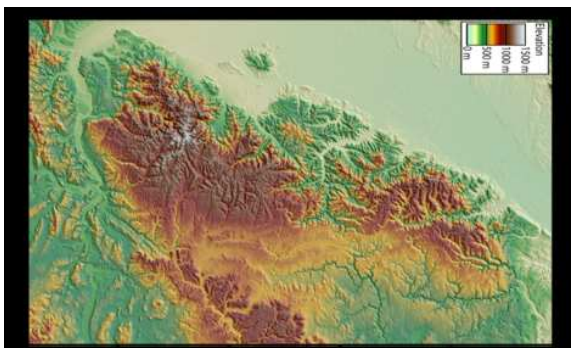

Figura 2.5: Model digital de la topografia de la Selva Negra (Baden-Wurtemberg, Alemanya).

2.3.2. Model d'elevacions de Catalunya

En aquest projecte, s'utilitzen mapes DEM de Catalunya facilitats per l'Institut Cartogràfic de Catalunya (ICC). Aquests, són models de malla regular que contenen altituds ortomètriques distribuïdes segons una quadrícula de 30 metres de costat, referits al sistema geodèisc de referència oficial ED50 i en projecció UTM.

El format dels fitxers .txt és el grid ASCII/ArcInfo. Hi ha una capçalera amb el nombre de columnes, nombre de files, coordenades del centre de píxel sud-oest, pas de malla i símbol de cota inexistent. A continuació vénen les cotes per files de Nord a Sud, i cada fila de Oest a Est. Es cobreix tota Catalunya en dos fitxers: un per a la meitat est i l'altre per l'oest. La zona que cobreixen cadascun dels dos fitxers és la següent:

```
CATA30_OEST_REV1 .TXT
VERTEX SW : 257995.00,   4484975.00
VERTEX NE : 396505.00,   4752005.00
EQUIESPAIAT:        30.00 m
NUM. FILES   :    8902
NUM. COLUMNES:  4618

CATA30_EST _REV1.TXT
VERTEX SW :    396505.00,   4484975.00
VERTEX NE :    535015.00,   4752005.00
EQUIESPAIAT:        30.00 m
NUM. FILES   :   8902
NUM. COLUMNES:  4618
```

2.3.3. Implementació

En l'eina de planificació de vol hi ha un objecte encarregat del processat dels fitxers en format grid ASCII/ArcInfo. S'emmagatzema tota la informació de la capçalera i tota la matriu d'altures. Al ser fitxers tan grans (cadascun pesa uns 300 Kbytes), l'objecte necessita força temps per carregar-los a memòria. No obstant un cop carregat tot el fitxer mai es manipula tot sencer. Si, per exemple, l'eina necessita la informació de 9 kilòmetres quadrats de terreny, l'objecte els selecciona per només treballar amb una petita part de la malla i així fer el codi més eficient.

Com s'ha dit anteriorment, aquest objecte està preparat per llegir qualsevol model digital del terreny en format grid ASCII/ArcInfo, per tant, si obtenim les dades d'elevació del terreny de qualsevol altra extensió de la Terra en aquest mateix format el mateix objecte ens les processaria però s'hauria de tenir en compte a quin sistema geodèsic de referència estan referides i en quina projecció.

2.4. L'algorisme de Bresenham

2.4.1. Introducció

També conegut com l'algorisme de la recta, l'algorisme de Bresenham és capaç de determinar quins punts d'una trama n-dimensional han de ser seleccionats amb la finalitat de formar una aproximació a una línia recta entre dos punts donats. Aquest algorisme va ser desenvolupat per Jack E. Bresenham l'any 1962 [6].

Un exemple gràfic del funcionament de l'algorisme de Bresenham és el presentat a la figura 2.6.

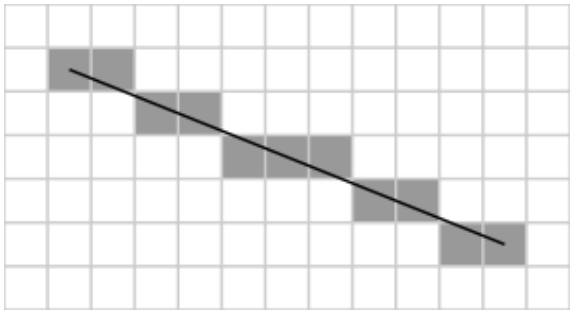

Figura 2.6: F:Exemple visual de la funcionalitat de l'algorisme de Bresenham

2.4.2. Aplicacions: L'algorisme de Bresenham com validor de rutes

Aquest algorisme té dues aplicacions molt importants a l'eina de planificació de vols ja que ens permet comparar l'alçada de vol de l'aeronau amb l'elevació del terreny. Aquest fet ens permet, primerament, trobar els obstacles que s'interposin en la ruta de l'aeronau i poder generar el diagrama de Voronoi corresponent per esquivar-lo, tal i com s'ha explicat en el tercer mode de funcionament del programa. D'altra banda ens permet purgar els diagrames de Voronoi tot reduint-ne el nombre d'arestes i nodes per així arribar a un temps d'execució acceptables, cosa que s'explicarà amb més detall en la secció dels diagrames de Voronoi i en la última secció d'aquest capítol.

Aquestes dues aplicacions fan de l'algorisme de la recta una peça clau dins de l'eina de planificació de vol per UAVs.

2.4.3. Implementació

Com es tracta d'una part molt important i ben diferenciada de les demés, s'ha creat un objecte propi que implementa l'algorisme. Aquesta classe només implementa un mètode que permet, donat dos punts en coordenades UTM referides a ED50[3], el model d'elevacions a utilitzar i l'alçada de vol de l'aeronau, verificar que en la recta que uneix aquests dos punts no s'hagi interposat cap obstacle i en cas de que hi hagi un conflicte amb l'elevació del terreny, indicar la posició exacta del mateix.

[3]Que les coordenades estiguin referides a aquest sistema geodèsic de referència és degut a que el model d'elevacions de Catalunya també està referit a aquest sistema de referència. En cas d'utilitzar un altre model, les coordenades han d'estar referides al sistema de referència del model.

2.5. Obtenció dels obstacles. Les llibreries GDAL

2.5.1. Introducció

Les llibreries GDAL (Geospatial Data Abstraction Library) tenen com a funció principal la lectura i escriptura de dades geoespacials publicada sota la llicència *X/MIT style Open Source* [8] per part de la fundació geoespacial de codi obert (Open Source Geospatial Foundation) [7].

2.5.2. Obtenció de les corbes de nivell

Una de les moltes utilitats que tenen aquestes llibreries és la de conversió de fitxers raster en vectorials. D'aquesta manera, podem obtenir directament les corbes de nivell a la cota desitjada a partir d'un model digital d'elevacions. L'obtenció de les corbes de nivell s'utilitza en el segon i tercer mode de funcionament de l'eina de planificació de vol. En el segon mode de funcionament, s'obtenen les corbes de nivell a una cota igual a l'altura de vol de l'aeronau en el marc territorial indicat per posteriorment seleccionar-ne els obstacles i així obtenir-ne els obstacles que generaran el diagrama de Voronoi. En el tercer mode de funcionament, quan s'executa l'algorisme de Bresenham i ens diu que hi ha un obstacle a la ruta en unes coordenades concretes es busquen les corbes de nivell a l'alçada de vol en una superfície de 9 kilòmetres quadrats al voltant del punt de conflicte.

A més, aquestes llibreries permeten la conversió entre els formats .shp (format del fitxer les corbes de nivells obtingut del model d'elevacions) a .kml (format utilitzat per Google Earth). Així doncs, podem representar sense cap problema les corbes de nivell de tot Catalunya amb Google Earth.

Les corbes de nivell que estan contingudes en el fitxer .kml generat per GDAL venen representades com a segments de recta. Per definir un segment de recta en Google Earth només és necessari donar el dos parells de coordenades que representen els punts extrems de l'aresta. El nivell de detall amb que GDAL obté les corbes de nivell supera amb creus el necessari per obtenir el diagrama de Voronoi resultant. Per tant, podríem pensar que si utilitzem tots els punts extrems dels segments de recta que conformen les corbes de nivell com a obstacles obtindríem un diagrama de Voronoi més acurat, i és així. No obstant, apareix un problema força important. Quant més obstacles siguin els que generin el diagrama de Voronoi més temps d'execució requerirà l'eina per calcular tots els algorismes. A més, com que utilitzem l'algorisme de Bresenham per descartar totes aquelles arestes que intersectin amb el terreny, el nostre diagrama de Voronoi resultant ja serà prou acurat, si més no totalment segur.

Per tant, per generar el diagrama de Voronoi resultant es segueix un procès de selecció uniforme d'obstacles basat amb els criteris de la taula 2.2:

La figura 2.7 és un exemple d'una corba de nivell sense realitzar-ne la selecció anterior. La figura 2.8 és la mateixa corba que la figura anterior però seleccionant només alguns dels seus punts.

Nombre de punts de la corba de nivell original(N)	Nombre de punts selecionats
$N > 10000$	300
$10000 < N < 1000$	100
$1000 < N < 100$	30
$100 < N < 10$	10
$10 < N < 1$	1

Taula 2.2: Criteris de selecció de punts de les corbes de nivell com a obstacles del diagrama de Voronoi.

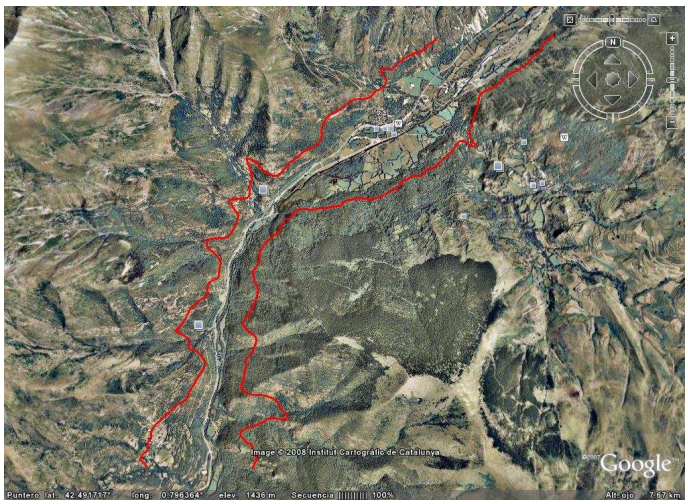

Figura 2.7: Exemple de corbes de nivell a una cota de 1200 metres de la regió de Barruera a l'Alta Ribagorça tal i com les genera GDAL

2.5.3. Implementació

Per utilitzar aquestes llibreries s'ha optat per fer una crida externa mitjançant la consola de Windows. De la conversió de fitxers raster en vectorials s'encarrega el mateix objecte encarregat de llegir el model digital d'elevacions. D'altra banda, de la selecció de punts com a obstacles del diagrama de Voronoi s'encarrega un objecte diferent.

2.6. Els diagrames de Voronoi

2.6.1. Introducció

Abans d'introduir els principals conceptes i característiques d'un diagrama de Voronoi és de vital importància fer una breu ressenya al món de la geometria computacional.

La geometria computacional, disciplina batejada per Shamos l'any 1975, poc coneguda en general donat el seu escàs temps d'existència i la seva escassa difusió, neix com a resposta a un gran nombre d'aplicacions i problemes geomètrics que necessiten d'algorismes eficients per a la seva resolució. En poques paraules es podria definir la geometria

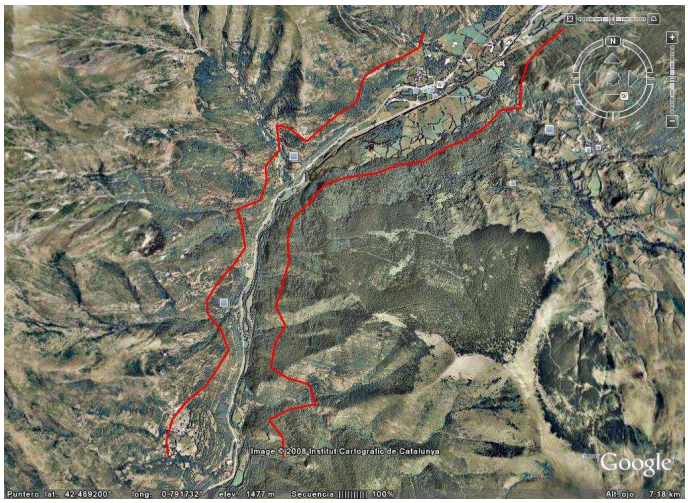

Figura 2.8: Exemple de corbes de nivell a una cota de 1200 metres de la regió de Barruera a l'Alta Ribagorça després del procés de selecció.

computacional com el nexe d'unió entre la geometria abstracta i el món de la computació.

El món de la geometria computacional és molt extens i divers. Entre els temes que tracta podem trobar des de problemes de triangulació fins a d'altres de proximitat, passant per problemes de localització entre d'altres. Doncs bé, molts d'aquests problemes, d'entre ells el que es tracta específicament més endavant, requereixen dels diagrames de Voronoi per resoldre'ls malgrat que, a primer cop d'ull, no tenen gaire coses en comú [9].

2.6.2. Definició formal

Sigui $P = \{ p_1, p_2, ..., p_n \}$ un conjunt de punts en un pla bi-dimensional euclidià. Aquests punts són els llocs (del anglès *sites*). Si ara dividim el pla assignant a cada punt del mateix pla amb el seu lloc més proper, tots els punts assignats a p_i formen la *regió de Voronoi* $V(p_i)$. Per tant, $V(p_i)$ és la regió formada pels punts de pla més propers a un lloc p_i que a qualsevol altre lloc [9].

$$V(p_i) = \{ x : |p_i - x| \leq |p_j - x| \, \forall j \neq i \} \tag{2.1}$$

Com es pot veure en la definició anterior, alguns punts no tenen un únic lloc més proper, un *únic veí* més proper. El conjunt de punts que tenen més d'un veí més proper son els que formen el *diagrama de Voronoi* V per al conjunt de llocs donat.

Un diagrama de Voronoi típic, és el que es mostra en la figura 2.9.

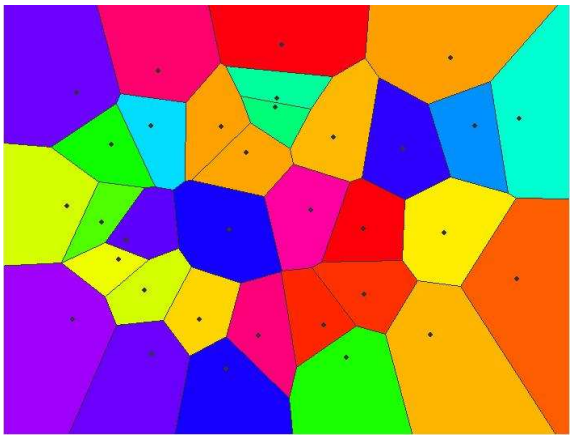

Figura 2.9: Diagrama de Voronoi per un conjunt de sites donat.

2.6.3. Motivació

Com s'havia proposat en la introducció, l'objectiu d'aquest projecte és la creació d'una eina de planificació de vol per UAVs partint d'uns obstacles. Ara bé, si substituïm els punts p_i per les coordenades dels obstacles obtenim un diagrama format pels punts que sempre estan el més lluny possible de dos obstacles donats, passant entre ells. En poques paraules, obtenim un diagrama en brut de possibles rutes més segures però alhora eficients de volar entre obstacles puntuals.

2.6.4. Implementació

Per la implementació s'ha triat l'algorisme de Fortune per ser precisament el més ràpid. En el pitjor dels casos l'algorisme de Fortune té una complexitat de $O(nlogn)$ que, en comparació amb la implementació emprant un algorisme incremental ($O(n^2)$) és força menor. A grans trets, l'algorisme de Fortune és del tipus *plane sweep algorithm*, és a dir, un algorisme d'escombrat del pla. Consisteix en fer passar una línia d'escombrat per sobre del pla deixant al costat escombrat la part del problema ja resolt i a l'altra la part que falta per resoldre'l [9].

En aquest projecte s'ha utilitzat una implementació de l'algorisme de Fortune ja realitzada en Csharp sota la llicència LGPL [10].

2.7. L'algorisme de cerca de camí òptim

2.7.1. Introducció

En poques paraules, un algorisme de cerca és aquell que pren un problema com a entrada i el resol mitjançant un exploració de l'espai de possibles solucions.

Hi ha diferents tipus d'algorismes de cerca, entre d'altres el que s'utilitza en aquest projecte que és del tipus *graph traversal*, que es podria traduir com recorregut del graf. Aquest tipus d'algorismes tracten de visitar tots els nodes d'un graf d'una determinada manera. Primerament, en aquesta secció es definirà què és un graf i per què els diagrames de Voronoi es poden considerar grafs sota les condicions sobre les quals l'estem utilitzant [11].

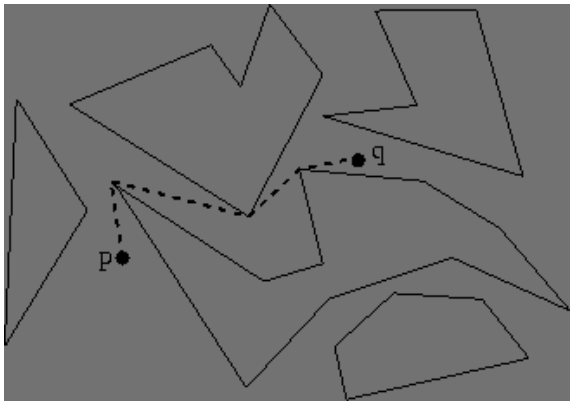

Figura 2.10: Exemple de cerca de camí òptim aplicat a la robòtica.

2.7.2. Els grafs

En Matemàtiques i en Computació, un graf és un conjunt d'objectes anomenats vèrtexs o nodes, units per arestes o arcs que permet establir relacions binàries entre ells [12].

En el cas del diagrama de Voronoi, es poden considerar els nodes d'un graf els vèrtexs del mateix diagrama i les arestes les unions entre els propis vèrtexs. Per tant la relació binària que s'estableix entre els nodes és el fet de que si es pot viatjar d'un node cap a un altre qualsevol. Aquesta relació s'estableix mitjançant les arestes.

A més a més es pot definir un pes o cost a cada aresta, de tal manera que viatjar d'un node a un altre que estiguin connectats tingui un cost. Aquest pot ser distància, cost de combustible, etc..

2.7.3. La cerca del camí òptim

Un cop tenim el graf ben definit no tenim res gaire diferent que un mapa de carreteres; els nodes farien la funció de ciutats i les pròpies carreteres d'arestes i la distància entre ciutats seria anàloga al pes de les pròpies arestes. Per tant, ja podem aplicar un algorisme de cerca de camí òptim, que tingui com a objectiu trobar el cost mínim entre un node d'origen i tots els altres nodes del graf.

2.7.4. Implementació

Per implementar aquest algorisme s'havia pensat en aplicar directament el de Dijkstra però aquest algorisme només es pot aplicar directament si els pesos de les arestes són constants. En el nostre cas, els pesos de les arestes vénen donats per la distància entre els dos nodes extrems de la mateixa i l'angle de gir entre aquesta aresta i l'anterior. Per tant, el pes de la nova aresta sempre dependrà del camí seguit anteriorment, no de l'aresta mateixa, així que no ha sigut possible implementar directament l'algorisme de Dijkstra. No obstant, com a alternativa, l'exploració del graf es realitza mitjançant un algorisme recursiu que s'atura quan es detecten cicles.

Com que les tasques que realitza l'eina en aquest apartat són importants i estan ben diferenciades respecte les anteriors s'ha creat un objecte exclusiu per realitzar aquestes funcions excepte el càlcul del pes. D'aquesta última tasca s'encarrega una classe abstracta que té implementada tres classes derivades de la primera. Una d'elles s'encarrega de calcular el pes que li assigna a cada aresta només tenint en compte la distància entre nodes. La segona només té en compte els angles entre arestes mentre que la tercera pondera les dos anteriors en uns factors a triar per l'usuari.

2.8. Viabilitat dels angles de gir

2.8.1. Introducció

No tots els camins possibles entre els nodes del graf resultant del diagrama de Voronoi són realitzables per una aeronau. Resulta físicament impossible que una aeronau realitzi un gir de 180 graus en un node. Per tant, s'ha de determinar, a partir de les característiques físiques de l'aeronau quin és el seu radi mínim de gir. Aquest és el tema que es desenvolupa en la aquesta secció.

2.8.2. Descripció física

La relació entre velocitat lineal i radi de gir d'una massa puntual ve donada per la següent equació newtoniana.

$$a_c = v^2/R \tag{2.2}$$

On a_c és l'acceleració centrípeta de l'aeronau. Si considerem l'aeronau com una massa puntual que està girant amb un angle de gir ϕ, llavors la força de sustentació L es pot descompondre en aquestes dos forces:

$$L_y = Lcos\phi L_x = Lsin\phi \tag{2.3}$$

La primera, L_y, ha de compensar el pes per mantenir l'altura de vol, metre que la segona L_x és l'encarregada de fer girar a l'aeronau. Per tant:

$$L_y = Lcos\phi = mgL_x = Lsin\phi = ma_c \tag{2.4}$$

Ara bé, si dividim L_x entre L_y obtenim la següent relació:

$$L_x/L_y = tan\phi = a_c/g \tag{2.5}$$

Així doncs, si substituïm el valor de a_c a aquesta última expressió obtenim el següent:

$$tan\phi = v^2/(Rg) \longrightarrow R = v^2/gtan\phi \tag{2.6}$$

Finalment podem observar que el radi de gir d'una aeronau en condicions ideals (especialment absència de vent) és proporcional al quadrat de la velocitat de l'aeronau i inversament proporcional a la tangent de l'angle de gir. Per tant, si volem obtenir el radi mínim de gir hem de maximitzar l'angle de gir i minimitzar la velocitat.

A més de la relació anterior, s'ha de tenir en compte el temps que tarda l'aeronau en passar d'una angle de gir de 0^o (vol en línia recta) a l'angle de gir màxim. Aquesta temps no es modela físicament sinó seguint unes pautes publicades per la OACI[4] en el document 4444 (Aircarft Operations).

Un cop tenim modelat el radi mínim de l'aeronau, aquest s'ha de comparar amb el gir que ha de realitzar. Primerament s'ha de tenir en compte que l'aeronau no pot realitzar el gir instantàniament, sinó que l'ha de començar a realitzar abans d'arribar al node. Així doncs, s'ha d'establir una distància màxima al node, d_{max} des de la qual es comenci a realitzar el gir. El gir a realitzar ha de ser tangent a les dues arestes perquè l'aeronau, un cop hagi finalitzat estigui ben orientat, en direcció al següent node tal i com s'il·lustra a la figura 2.11:

Es demostra geomètricament que el radi òptim de gir és el següent:

$$R = d_{max}tan(\phi/2) \tag{2.7}$$

En la figura anterior tenim que l'angle de gir és de 90 graus per tant $R = d_{max}$. En resum, si el radi mínim de gir de l'aeronau és superior a R aquesta NO podrà efectuar el viratge. En cas contrari, si.

2.8.3. Implementació

De la implementació del càlcul del radi mínim de gir s'encarrega l'objecte responsable d'inicialitzar l'aeronau mentre que el càlcul del radi necessari per efectuar el gir sobre un

[4] Organització d'Aviació Civil Internacional

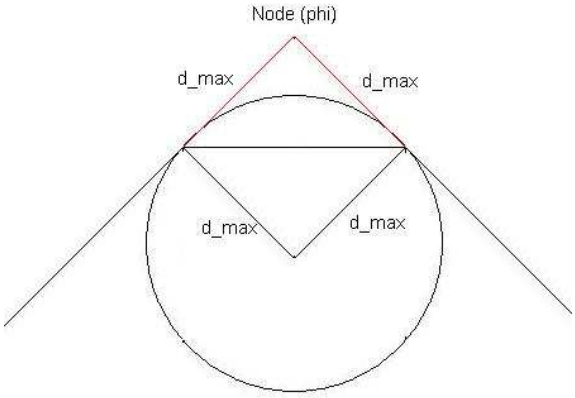

Figura 2.11: Angle de gir per a phi = 90º.

node determinat s'encarrega l'objecte responsable de calcular els pesos sobre les arestes. La variable d_{max} la ha d'introduir el propi usuari. Per la seva determinació s'han de tenir en compte diferents aspectes com la orografia del terreny o les característiques pròpies de l'aeronau. Quant més maniobrabilitat disposi l'aeronau més petit tindrà el radi mínim de gir i, per tant, podrà efectuar girs més tancats o, per un mateix angle de gir, no necessitarà valors grans de d_{max}. S'ha de tenir en compte que quant més gran sigui el valor de d_{max}, menys fidel serà la trajectòria de l'aeronau al graf de Voronoi.

S'ha de tenir en compte que les unitats de totes les variables definides en aquesta secció són del Sistema Internacional.

2.9. Obtenció de rutes vàlides mitjançant obstacles no puntuals

2.9.1. Introducció

En el segon i tercer mode de funcionament de l'eina de planificació de vol per UAVs, els obstacles que després definiran el graf que l'aeronau podrà sobrevolar no són puntuals. En efecte, la utilització de les corbes de nivell provoca la necessitat d'anar un pas més enllà en quant a l'obtenció d'obstacles es refereix.

Utilitzant el mateix codi que ens genera els diagrames de Voronoi però combinant-lo amb l'algorisme de Bresenham podem obtenir rutes vàlides utilitzant obstacles no puntuals com ara les corbes de nivell generades per GDAL.

2.9.2. Implementació

Ja s'ha explicat a la secció de les llibreries GDAL com s'obtenen els obstacles a partir de les corbes de nivell. Un cop es tenen les corbes de nivell, s'obté el diagrama de Voronoi mitjançant l'algorisme de Bresenham. No obstant, fins aquí, el resultat obtingut deixa molt que desitjar. En les figures 2.12 i 2.13 es mostren uns exemples dels possibles diagrames de Voronoi obtinguts.

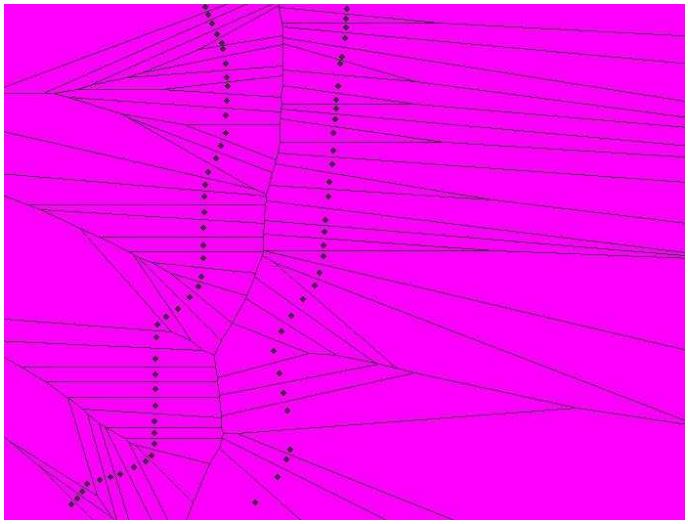

Figura 2.12: Exemple de diagrama de Voronoi generat a partir d'un applet.

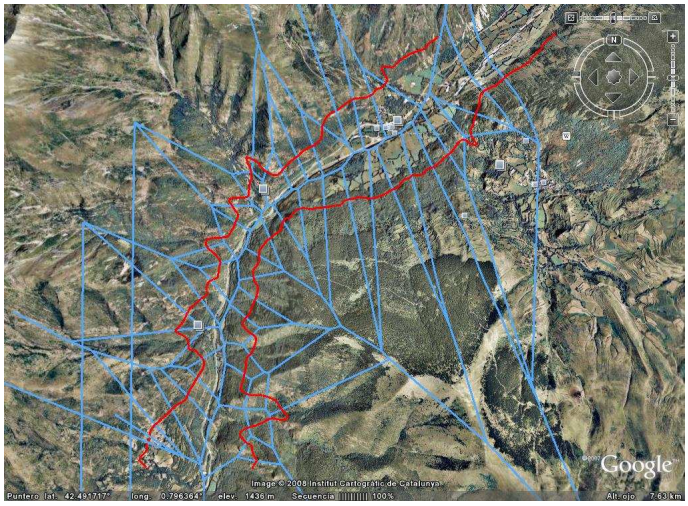

Figura 2.13: Exemple de diagrama de Voronoi generat a partir de corbes de nivell obtingudes amb GDAL.

Com es pot observar, és necessària una eina que permeti purgar el diagrama obtingut tot eliminant aquelles arestes que no siguin útils. Resulta trivial que si les corbes de nivell estan generades a l'altura de vol de l'aeronau, qualsevol aresta que les intersequi hauria de ser eliminada, perquè de ben segur que interseca amb el terreny.

Hi ha un algorisme que ja està implementat en aquest codi que s'adequa perfectament a aquesta necessitat de purgació: l'algorisme de Bresenham. Per purgar el diagrama,

utilitzarem l'algorisme de l'equació de la recta com a validor de rutes, tal i com s'ha explicat anteriorment. Així, ens assegurarem d'eliminar totes les arestes que no ens interessin i obtindrem un digrama com el de la figura 4.7.

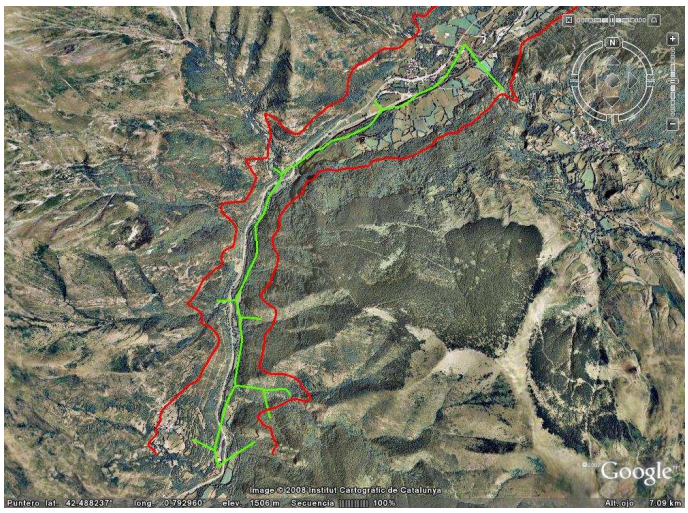

Figura 2.14: Diagrama de Voronoi purgat.

D'aquesta manera alleugerirem la càrrega de treball a l'algorisme de cerca de camí òptim, que ja no haurà de buscar rutes per camins que igualment l'aeronau no podria sobrevolar.

CAPÍTOL 3. DESCRIPCIÓ I ORGANITZACIÓ DEL CODI

3.1. Introducció

En aquest capítol es pretén donar una visió general de l'organització del codi que forma l'eina de planificació del vol. En primer lloc s'explicaran breuement les eines de programació utilitzades per endinsar-nos posteriorment en la organització dels objectes que formen el propi codi. Es descriurà detalladament quines són les responsabilitats de cada classe i com interactuen entre elles.

3.2. L'eina de programació utilitzada. Csharp

3.2.1. Introducció

Csharp és un llenguatge desenvolupat per Microsoft a finals del segle passat. Forma part de la plataforma .NET i es tracta d'un llenguatge orientat a objectes, simple, segur, modern, d'alt rendiment i amb especial èmfasi en Internet i els seus estàndards (com XML) [13].

3.2.2. La plataforma .NET

La plataforma .NET és una plataforma de desenvolupament de software que prioritza la creació ràpida d'aplicacions, la independència del llenguatge i la transparència a través de xarxes.

La formen les següents parts:

- Un conjunt de llenguatges de programació (Csharp, Jsharp, C++ gestionat...)
- Un conjunt d'eines de desenvolupament (Monodevelop, Visual Studio .NET)
- Una llibreria de classes àmplia i comuna per a tots els llenguatges.
- Un sistema d'execució de llenguatge comú (CLR[1])
- Un conjunt de servidors .NET
- Un conjunt de serveis .NET

[1] CLR:Common Language Runtime

Els principals punts forts de la plataforma són els següents:

- **Independència en el llenguatge:** Tots els llenguatges que conformin amb els estàndards .NET podran interoperar entre si d'una forma totalment transparent. Les classes podran ser heretades entre llenguatges, per exemple.
- **Llibreria de classes comuna:** Més de 4000 llibreries, classes i mètodes inclosos en la plataforma .NET estan disponibles per a tots els llenguatges.
- **Multiplataforma:** Quan un programa és compilat no és compilat directament en un arxiu executable sinó en un llenguatge entremig anomenat IL[2] el qual podrà ser executat en qualsevol plataforma on el CLR estigui disponible.

3.2.3. Csharp

Encara que és possible escriure codi per la plataforma .NET en molts altres llenguatges, Csharp és l'únic que ha sigut dissenyat específicament per ser utilitzat en aquesta, per la qual cosa programar-la mitjançant Csharp és molt més senzill i intuïtiu que amb qualsevol altre d'aquests llenguatges ja que Csharp no té elements heretats innecessaris en .NET. Per aquesta raó es sol dir que Csharp és el llenguatge nadiu de .NET.

La sintaxi i estructuració de Csharp és molt similar a la de C++, ja que la intenció de Microsoft amb Csharp és la de facilitar la migració de codis i l'aprenentatge als desenvolupadors habituats a ells.

Un llenguatge que hagués sigut un també interessant per escriure codi per la plataforma .NET és Java però els problemes amb l'empresa creadora del mateix, Sun, Microsoft un nou llenguatge que afegís a les provades virtuts de Java les modificacions que Microsoft tenia pensat afegir-hi i fer-lo un llenguatge orientat al desenvolupament de components.

3.3. L'eina de visualització utilitzada. Google Earth

3.3.1. Introducció

Google Earth és un software similar a un sistema d'informació geogràfica, creat per l'empresa Keyhole Incorporated que permet visualitzar imatges en tres dimensions del planeta, combinant imatges de satèl·lit, mapes i el motor de cerca Google [14].

3.3.2. Representació espaial amb Google Earth. Els fitxers kml

Un dels punts forts de Google Earth és la possibilitat de representació de dades geogràfiques mitjançant fitxers amb un format especial, el format kml. KML utilitza una

[2] De l'anglès, Intermediate Language

estructura d'etiquetes amb elements i atributs basada en l'estàndard XML.

Com que en aquest projecte s'utilitza Google Earth per a la representació dels resultats finals, el codi té funcions encarregades d'escriure els resultats en format KML per a la seva posterior visualització.

Figura 3.1: Exemple del potencial de representació geogràfica de Google Earth.

3.4. Organització del codi

3.4.1. Introducció

Com qualsevol codi escrit en Csharp, aquest està organitzat en diferents classes que interactuen les unes amb les altres. Cada classe té atribuïdes unes funcions o responsabilitats que han de realitzar a partir de les dades proporcionades per l'usuari o pels propis resultats d'altres classes.

Així doncs, a continuació s'explicarà quina és exactament la funció de cada classe, qué és el que necessita per part de l'usuari o d'altres classes i quins resultats aporta.

Les classes són les següents, per ordre alfabètic:

- **CAcalculapes**
- **Cadj**
- **Caeronau**
- **Cangle**
- **CArcInfo**
- **Cbresenham**
- **Ccomposada**

- **Cconversor**
- **Cdist**
- **Cin**
- **Ckmlcalib**
- **Cout**
- **Cpath**
- **Cvoronoi**

3.5. El codi classe a classe

3.5.1. CAcalculapes

Aquesta classe és de tipus abstracta. Hi ha ocasions, quan es desenvolupa una jerarquia de classes, en que hi ha un comportament que està present en totes elles però es materialitza de diferent forma. Per exemple, podríem tenir una classe anomenada FiguraGeometrica i una sèrie de classes que podrien ser Cercle, Pentàgon, etc. Podria existir un mètode "dibuixar"donat que en totes les figures es pot realitzar aquesta acció, però les accions concretes per realitzar aquest mètode depenen del tipus de figura en concret (de la seva classe). A demés, l'acció 'dibuixar' no té sentit que sigui implementada per la classe FiguraGeometrica perquè aquesta representa una abstracció del conjunt de figures possibles.

En el cas en que ens ocupa, la classe CAcalculapes és abstracta perquè té la funció de calcular el pes que li pertoca a cada aresta però, com que hi ha diferents maneres de calcular-ho, no té sentit implementar un mètode que retorni el pes de l'aresta. En canvi, en aquesta classe es defineix què és el que es necessita (com a mínim) per calcular el pes d'una aresta. D'aquesta manera, podem derivar classes de forma jeràrquica d'aquesta en les quals si que està implementat el mètode que calcula el pes per a cada aresta, com ara la classe Cdist, la Cangle o la Ccomposada.

Aquesta classe necessita saber de la classe Cpath quins són els nodes visitats per l'algorisme de cerca de camí òptim que implementa el nombre de nodes visitats. Per altra banda necessita de la classe de Cadj la matriu d'adjacència del graf. Té com a objectiu (i per tant totes les derivades de la mateixa) retornar com a resultat el pes que se li ha d'assignar a la aresta en qüestió.

3.5.2. Cadj

Classe encarregada de crear la matriu d'adjacència d'un graf coneixent les coordenades de cada node i les seves respectives unions.

La classe Cadj és l'encarregada de crear la matriu d'adjacència[3] d'un graf.

Per poder crear-la, necessita saber quines són les coordenades dels nodes i la de les arestes[4] per poder comparar-les i crear-ne la matriu. Aquestes coordenades les obté de la classe Cvoronoi. Com a resultat retorna la pròpia matriu d'adjacència del graf de Voronoi.

3.5.3. Caeronau

Classe encarregada de calcular el radi mínim de gir dels UAVs a partir de l'angle màxim de 'alabeo' i la velocitat de la pròpia aeronau. Aquest radi es compara amb el necessari per efectuar l'angle de gir en qüestió. Si el radi mínim de gir és superior al radi necessari per realitzar el gir, és a dir, l'aeronau és incapaç de realitzar el gir proposat, la classe Cangle s'encarrega de retornar un pes infinit per l'aresta que se li està demanant el pes. En altres paraules, la classe CAeronau és la classe encarregada de dir-li a la classe CAngle si l'angle de gir en qüestió és viable per l'aeronau o no.

3.5.4. Cangle

Classe derivada de la classe abstracta CAcalculapes. Té com a funció retornar el pes de l'aresta tenint en compte únicament l'angle definit pels dos últims segments del path.

A més de les dades que necessita per ser una classe derivada de CAcalculapes, necessita saber quin és el radi mínim de gir de l'aeronau (obtingut de la classe CAeronau) la distància màxima al node on l'aeronau pot començar a realitzar el viratge (obtingut directament de l'usuari) i les coordenades dels nodes, per poder calcular els angles de gir entre arestes (obtingudes de la classe Cvoronoi).

3.5.5. CArcInfo

Aquesta classe és l'encarregada de la manipulació dels fitxers DEM. Té una estructura on es guarda tota la informació del DEM i es capaç de llegir-lo, escriure'l, llegir només una part i crear els fitxer .shp per la posterior obtenció de les corbes de nivell.

La classe CArcInfo no necessita cap resultat de cap altra classe però si que l'usuari ha d'indicar la ruta a l'arxiu que contingui el DEM.

[3] Una matriu d'adjacència és una matriu quadrada que s'utilitza per representar relacions binàries. En el cas d'un graf, indica si dos nodes estan units o no (1 = units ó 0 = no units).

[4] Es consideren les coordenades de les arestes les dels punts extrems de les mateixes

3.5.6. Cbresenham

Aquesta classe és l'encarregada d'implementar l'algorisme de la recta. Serveix per purgar el diagrama de Voronoi (ens indica quines són les arestes que estan per sota de l'altura de vol) i també ens indica on hi ha un obstacle en la ruta que l'aeronau segueix.

Com que aquesta classe es dedica a realitzar comparacions entre el mapa DEM i l'alçada de vol de l'aeronau necessita de la classe CArcInfo l'estructura on està emmagatzemada tota la informació del DEM. A més, necessita saber les coordenades (en format UTM referenciades a ED50) dels dos extrems de la recta que ha d'anar comparant amb el terreny per part de l'usuari o de la classe Cvoronoi i l'alçada de vol de l'aeronau. Com a resultats ens retorna un booleà indicant si la recta ha interceptat el terreny en algun punt i en cas de que sigui afirmatiu, les coordenades del primer punt d'intersecció. A més a més ens indica per quin extrem de la recta s'ha començat a aplicar l'algorisme.

3.5.7. Ccomposada

Classe derivada de la classe abstracta CAcalculapes. Té com a funció retornar el pes de l'aresta ponderant la distància entre els extrems amb l'angle que forma amb l'aresta anterior, és a dir, ponderant els pesos obtinguts amb les altres dues classes derivades implementades (Cdist i Cangle). Per fer-ho el constructor crea dos objectes, un de cada classe derivada anterior.

Com que aquesta classe pondera els resultats de les altres classes derivades necessita aquests mateixos. A més necessita per part de l'usuari els factors de ponderació. Com a resultat, aquesta classe retorna el pes ponderat.

3.5.8. Cconversor

Aquesta classe és l'encarregada d'efectuar totes les possibles conversions entre els dàtums ED50 i WGS84 i entre les respectives projeccions UTM mitjançant les fórmules de Coticchia-Surace.

Com és evident, aquesta classe necessita com a entrada les coordenades dels punts a convertir i, depenent de la conversió també necessita els fusos UTM on es troben els punts en qüestió. Retorna els punts referits al nou sistema de referència i en el sistema de coordenades desitjat.

3.5.9. Cdist

Classe derivada de la classe abstracta CAcalculapes. Té com a funció retornar el pes de l'aresta tenint en compte únicament la longitud d'aquesta.

A demés de les dades que necessita per ser una classe derivada de CAcalculapes, ne-

cessita de la classe Cvoronoi les coordenades dels nodes per poder calcular la distància entre els mateixos. Retorna la distància entre els dos nodes en qüestió.

3.5.10. Cin

Aquesta classe és l'encarregada de manipular fitxers en format .kml i obtenir els obstacles per la posterior creació del diagrama de Voronoi.

El fitxer .kml que manipula pot haver estat generat per la classe Ckmlcalib o directament generat per l'usuari posant obstacles manualment mitjançant Google Earth. El resultat que dóna aquesta classe són les coordenades geogràfiques referides a l'el·lipsoide WGS84 dels obstacles per a la posterior creació del diagrama de Voronoi.

3.5.11. Ckmlcalib

Aquesta classe s'encarrega de generar un fitxer en format .kml del fitxer .shp generat per la classe CArcInfo i reduir-ne el nombre de punts que formen les corbes de nivell per que la classe Cin pugui generar el vector d'obstacles sobre els quals s'obté el diagrama de Voronoi.

Així doncs, la única cosa que necessita és la ruta a l'arxiu .shp i dóna com a resultat un fitxer .kml amb les coordenades dels obstacles.

3.5.12. Cout

Classe encarregada de crear fitxers en format .kml per a la posterior visualització dels resultats amb Google Earth. Com que aquesta classe té la responsabilitat de generar arxius amb els resultats necessita tota la informació que s'ha anat generant com ara les coordenades dels nodes de la classe Cvoronoi o els camins òptims a seguir per arribar a tots els punts del diagrama i les rutes dels arxius que vagi generant.

Normalment es generen els arxius següents:

- **Arxiu per la visualització d'obstacles**
- **Arxiu per la visualització de nodes del diagrama de Voronoi**
- **Arxiu per la visualització dels trams rectes fora dels diagrames de Voronoi**[5]
- **Arxiu per la visualització del camí òptim a seguir dins del diagrama de Voronoi**

[5] Els trams calculats directament per l'algorisme de Bresenham

3.5.13. Cpath

Aquesta classe és l'encarregada de trobar el camí més curt entre un node i tots els altres d'un graf coneixent la matriu d'adjacència del mateix i els pesos de cada aresta. Per tant implementa la funció recursiva de cerca de camí òptim. Dóna com a resultat una matriu que indica quin és el camí òptim a seguir (si existeix algun) node a node per arribar a tots els nodes del graf.

3.5.14. Cvoronoi

La classe Cvoronoi s'encarrega de, a partir de les dades obtingudes amb la classe Cin, processar el diagrama mitjançant l'algorisme de Fortune per poder realitzar correctament el posterior càlcul dels pesos de les arestes i processar la cerca del camí òptim.

Un cop el diagrama està generat i s'ha creat també la matriu d'adjacència mitjançant la classe Cadj, la classe Cvoronoi te una funció de purgació del diagrama, que és l'encarregada de cridar a la classe Cbressenham perquè purgui el diagrama eliminant les arestes que han interceptat la superfície.

Per tant, necessita de la classe Cin les coordenades dels obstacles i genera dos vectors, un amb les coordenades dels nodes i un altre amb les coordenades de les arestes que, òbviament, coincideixen.

CAPÍTOL 4. CASOS PRÀCTICS

4.1. Introducció

En aquest capítol es presenten tres casos pràctics del funcionament de l'eina de planificació de vol per UAVs. En el primer cas, serà el propi usuari qui introduirà, mitjançant Google Earth, les coordenades dels obstacles i es generarà el diagrama de Voronoi sobre les mateixes. En el segon cas, l'usuari introduirà l'àrea per on sobrevolarà l'aeronau. En el tercer cas, l'usuari introduirà directament al codi les coordenades d'origen i destí i posteriorment es generarà la ruta òptima entre aquets dos punts. En ambdós casos s'establirà una alçada de vol que s'introduirà directament al codi.

4.2. Primer cas

4.2.1. Qué es necessita de l'usuari?

Com ja s'ha esmentat a la introducció, en aquest primer cas l'usuari a d'introduir mitjançant Google Earth les coordenades dels obstacles sobre els quals l'aeronau no podrà volar. A més, es necessita la ruta a l'arxiu que contingui el mapa DEM i l'alçada de vol. En aquest cas pràctic, situat a la part més oriental dels Pirineus catalans, s'ha establert una alçada de vol de 3000 metres i la distribució dels obstacles és la que es veu a la figura 4.1.

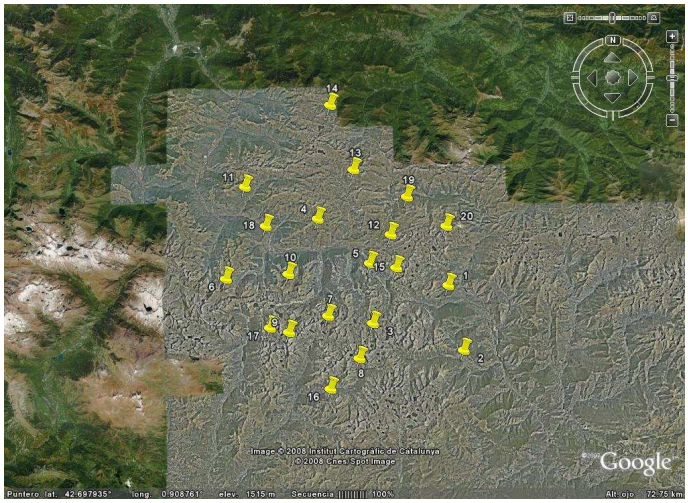

Figura 4.1: Distribució d'obstcles seleccionats per l'usuari.

4.2.2. Execució del programa

Un cop es tenen totes les dades, el programa calcula el diagrama de Voronoi per al conjunt d'obstacles donat. Abans de purgar-lo el diagrama té l'aspecte de la figura 4.2.

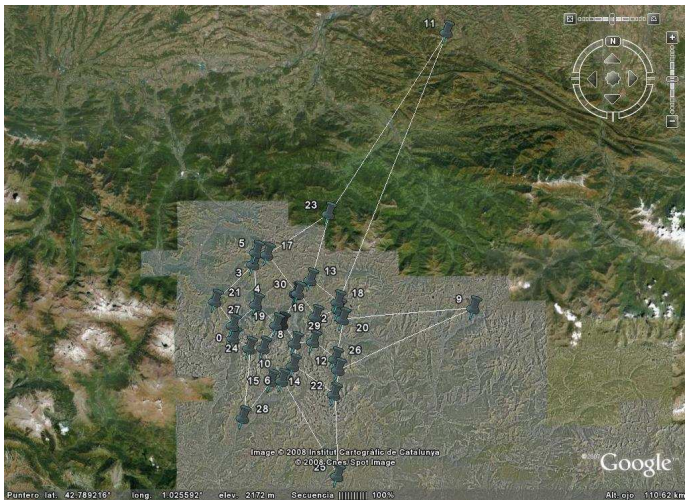

Figura 4.2: Diagrama de Voronoi pel conjunt d'obstacles donat, sense purgar.

Com es pot comprovar, és necessària una purgació del diagrama perquè, per exemple no tenim informació sobre l'elevació del terreny al voltant del node 11, ja que està situat a França, no a Catalunya. Un cop el programa purga el graf, s'obté la figura 4.3. Les xinxetes verdes representen els nodes del diagrama de Voronoi mentre que les de color groc són els obstacles. Els segments de recta vermells representen les arestes del diagrama.

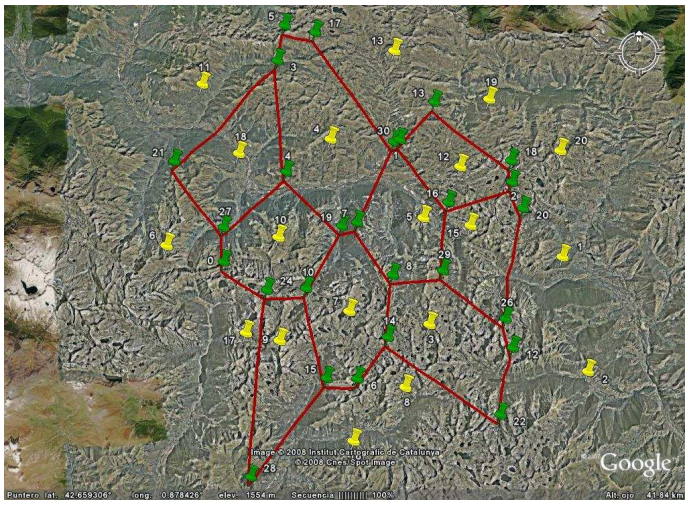

Figura 4.3: Diagrama de Voronoi pel conjunt d'obstacles donat, purgat.

Com s'ha comentat anteriorment, l'alçada de vol escollida per l'usuari és de 3000 metres, cosa que ha causa que la purgació del diagrama només es centrés en l'exclusió del graf d'aquells punts dels quals no tenim informació d'elevació. Si, per exemple, l'usuari hagués escollit una alçada de vol de 2750 metres tot mantenint els obstacles, el diagrama de Voronoi purgat hagués sigut el de la figura 4.4.

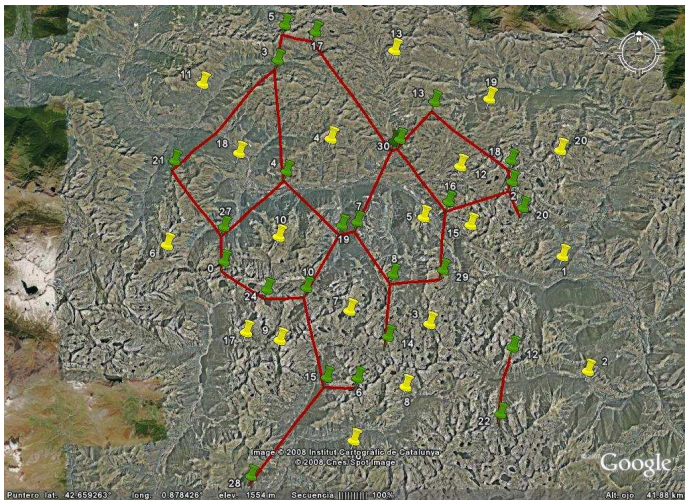

Figura 4.4: Diagrama de Voronoi pel conjunt d'obstacles donat, a una altura de vol de 2750 metres.

Com es pot observar, s'ha eliminat arestes per les quals l'aeronau ja no pot circular i fins i tot ara ja no es pot viatjar des d'un mateix node a tots els demés. Si encara reduíssim més l'alçada de vol la purgació del diagrama seria molt més severa. No obstant, seguirem treballant amb l'alçada de vol inicial, la de 3000 metres.

Un cop s'obté el diagrama purgat, l'usuari ha d'escollir quin node és el d'origen i quin és el de destí perquè el programa pugui calcular el camí òptim per realitzar aquesta ruta. En la figura 4.5 es presenta la ruta òptima per anar del node 21 al node 22, calculada pel programa.

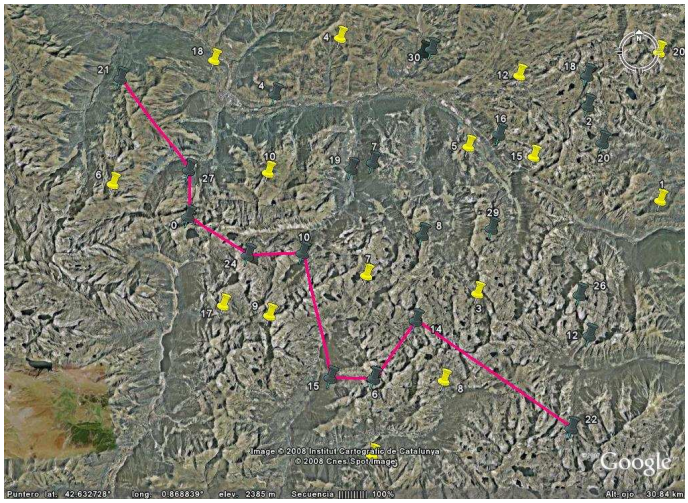

Figura 4.5: Camí òptim entre el nodes 21 i 22.

4.2.3. Conclusions

En aquest cas pràctic s'ha pogut observar un dels modes de funcionament de l'eina de planificació de vol per UAVs. Potser és el mode més senzill, però alhora és molt pràctic

quan ens limitem a una extensió territorial més aviat petita. La introducció manual dels obstacles utilitzant Google Earth és molt dinàmica y senzilla i el temps d'execució del programa es redueix considerablement.

4.3. Segon cas

4.3.1. Qué es necessita de l'usuari?

Com ja s'ha esmentat en la introducció, el segon cas pràctic l'usuari a d'introduir les coordenades dels extrems d'una regió territorial en format UTM referit a ED50 a més de l'alçada de vol de l'aeronau i la ruta a l'arxiu amb el model digital d'elevacions. En aquest cas pràctic s'ha escollit com a marc territorial un de molt proper a l'anterior, en el Pirineu català més occidental, a la Vall de Boí, a l'Alta Ribagorça. L'alçada de vol de l'aeronau serà de 1200 metres Al ser una vall, ens permetrà veure més destacadament el procediment que segueix el programa per determinar el camí òptim i sense obstacles entre els dos extrems de la vall. Com es pot comprovar, a diferència del cas anterior, l'usuari en cap moment ha d'indicar ell mateix els obstacles, sinó que és el propi programa qui, utilitzant les corbes de nivell obtingudes per les llibreries GDAL a partir del mapa DEM, genera els seus propis obstacles.

4.3.2. Execució del programa

Un cop l'usuari ha introduit en el propi codi les coordenades del marc territorial, el programa carrega el mapa DEM i selecciona les dades que l'interessen del mateix[1]. Un cop ha realitzat aquesta operació, crea les corbes de nivell a l'alçada de vol de l'aeronau, i n'obté els obstacles tal i com es veu en la figura 4.6:

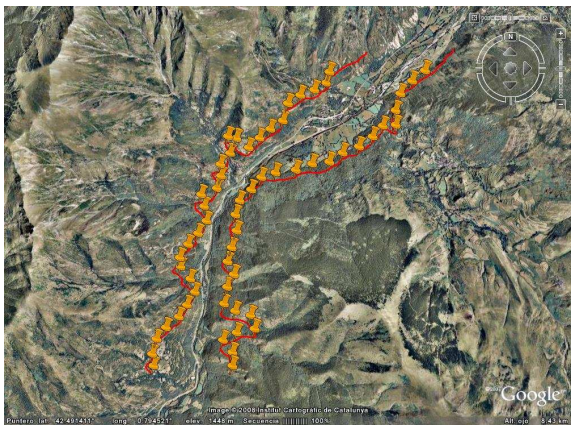

Figura 4.6: Corba de nivell a 1200 metres amb els obstacles creats.

[1]A partir de les coordenades introduïdes per l'usuari troba les elevacions dels punts dins del marc i descarta totes les demés per, tal i com s'ha comentat en capítols anteriors, millorar l'eficiència de l'eina

Ara que ja s'han obtingut els obstacles, el programa opera de forma anàloga a l'anterior cas però, es pot comprovar en la figura següent, que el graf obtingut després de la purgació del diagrama és encara més simple que l'anterior. Es considera la cerca del camí òptim entre el node situat a l'extrem sud del marc territorial i el situat a l'extrem nord, tal i com es pot observar en les figures 4.7 i 4.8

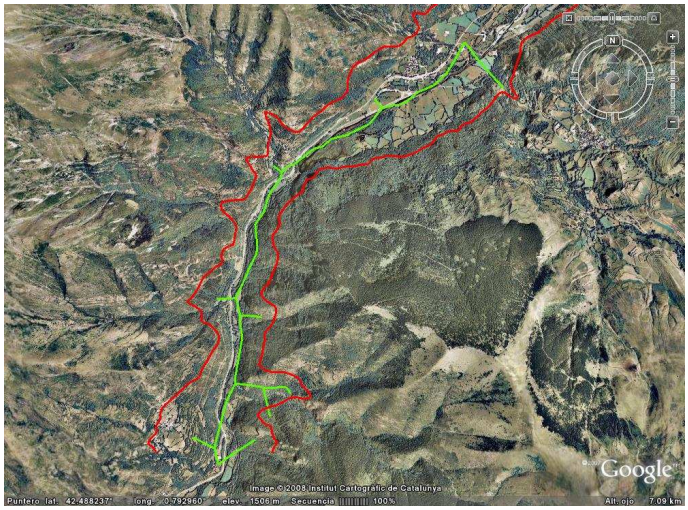

Figura 4.7: Diagrama de Voronoi purgat.

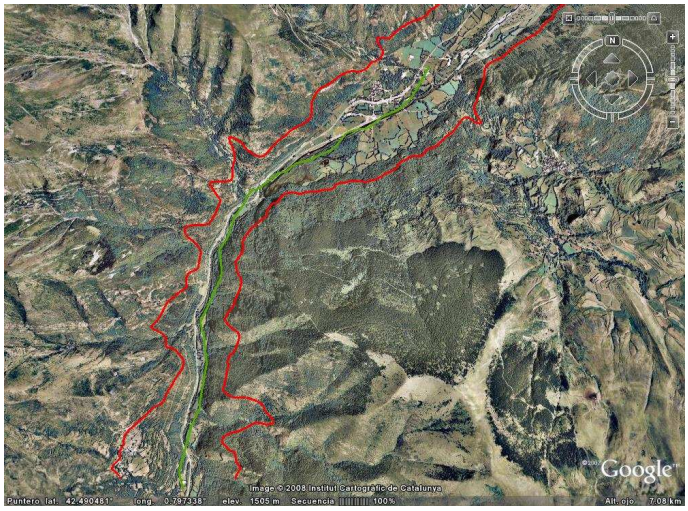

Figura 4.8: Camí òptim entre els nodes dels extrems sud i nord.

4.3.3. Conclusions

En aquest cas pràctic s'ha pogut observar un mode més avançat de funcionament de l'eina de planificació de vol per UAVs. Aquest mode té un avantatge molt clar respecte l'anterior. L'usuari ja no s'ha de dedicar a indicar a l'aeronau quins són els llocs que no es poden sobrevolar sinó que és el propi programa que indica a l'aeronau per on no ha de passar. Aquesta característica li dóna molt més dinamisme a l'eina.

4.4. Tercer cas

4.4.1. Qué es necessita de l'usuari?

Aquest últim cas, com ja s'esmenta a la introducció d'aquest capítol, consisteix en donar un pas més enllà i portar l'eina de planificació de vol per UAVs al mode més avançat. En el cas anterior, se li demanava a l'usuari un marc territorial a partir de dos parells de coordenades. En aquest últim mode no això no és necessari. A més d'introduir la ruta a l'arxiu que conté el model digital d'elevacions i l'alçada de l'aeronau, l'usuari només s'ha de preocupar de conèixer les coordenades d'origen o posició inicial de l'aeronau i destinació. El propi programa s'encarrega de seleccionar ell mateix els marcs territorials per salvar els obstacles i trobar la ruta òptima. En aquest cas l'alçada de vol és de 2550 metres i es consideren les coordenades geogràfiques (WGS84) per als punts d'origen i destí següents, situats altre cop a l'Alta Ribagorça:

`Punt inicial: Lat: 42.4997777º Lon: 0.9933333º`
`Punt de destí: Lat: 42.4977777º Lon 0.9480555º`

4.4.2. Execució del programa

Primerament, el programa tria el punt per on començarà a aplicar tot l'algorisme. En aquest cas ha triat el punt de destí. Posteriorment, tal i com s'ha explicat en capítols anteriors, el programa intenta arribar al punt de destí traçant una línia recta, com es pot observar en la figura 4.9.

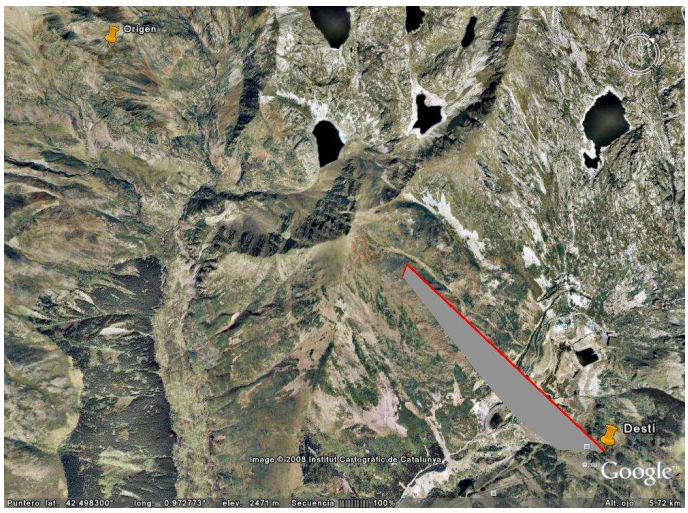

Figura 4.9: Primer tram recte de la ruta.

Un cop troba l'obstacle, busca la corba de nivell adient i genera el diagrama de Voronoi local corresponent. Llavors cerca el camí òptim entre el node més proper al punt que s'ha arribat mitjançant l'algorisme de la recta en el pas anterior i el node més proper al punt de destí. com es pot veure en la figura 4.10.

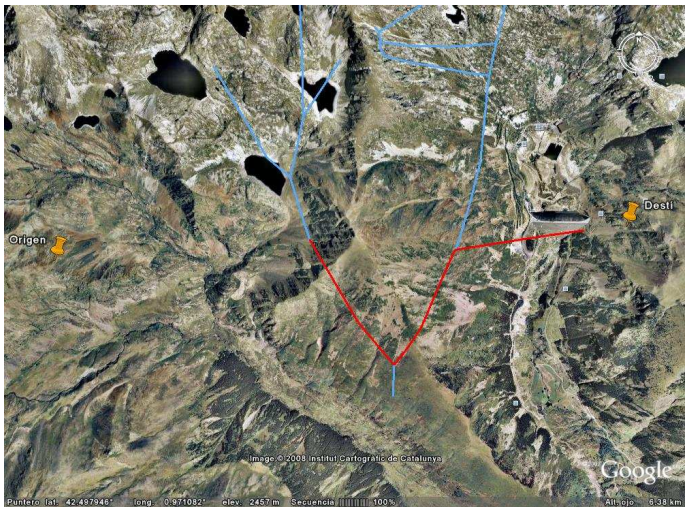

Figura 4.10: Diagrama de Voronoi generat per aquest cas amb ruta òptima triada pel programa.

Des del node de més proper al destí es torna a traçar una recta amb l'objectiu d'arribar amb la menor distància possible. En el nostre cas no hi ha cap altre obstacle entre el punt on ens trobem i el punt de destí així que no cal tornar realitzar els càlculs anteriors. El camí triat es presenta a continuació, a la figura 4.11:

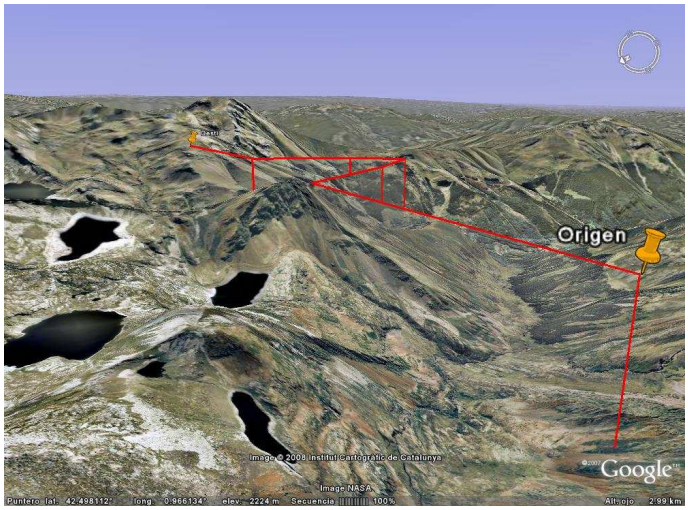

Figura 4.11: Camí triat per l'eina de planificació de vol per aquest últim cas.

4.4.3. Conclusions

En aquest cas pràctic s'ha posat en evidència el veritable mode de funcionament de l'eina de planificació de vol per UAVs. Malgrat que com es pot observar en la última figura, no sigui el camí més òptim entre els dos punts, si que s'aproxima. En aquest últim mode s'ha de tenir cura d'escollir bé l'alçada de vol perquè el seu bon funcionament és inversament proporcional a la mida dels obstacles, que alhora és inversament proporcional a l'alçada

de vol. Per altures de vol més baixes els obstacles (les corbes de nivell) augmenten en superfície i costa més trobar un camí vàlid per salvar l’obstacle.

CAPÍTOL 5. CONCLUSIONS

5.1. Assoliment d'objectius

L'eina desenvolupada en aquest projecte compleix els objectius que s'havien plantejat quan es va forjar l'idea de la seva el·laboriació i que s'han plasmat en la introducció d'aquesta mateixa memòria. El programa permet generar de manera off-line les dades necessàries per la creació d'un pla de vol a alçada constant de manera automàtica. Els tres modes de funcionament li donen a l'eina una versatibilitat considerable i la no necessitat d'haver d'executar tota l'eina quan això no sigui necessari pel cas en que s'estigui treballant.

A més, cal dir que l'elecció com a entorn de programació del llenguatge Csharp ha sigut del tot encertada perquè ha facilitat, sobretot la legibilitat del codi a persones alienes al projecte i la programació per part meva. També ha sigut important l'elecció del Google Earth com a mitjà de representació dels resultats obtinguts ja que dóna una visió molt entenedora del que està realitzant en cada moment l'eina i, a més, Google Earth és un software força proper a tothom.

5.2. Aplicacions

Hi ha diverses aplicacions a tenir en compte, algunes es comenten a continuació.

El segon mode de funcionament de l'eina pot ser perfectament aplicat a la prevenció d'incendis en valls abruptes i de difícil accés mitjançant UAVs. Simplement disposant d'un mapa d'elevacions mitjanament precís, s'elabora un marc territorial de vigilància i s'indica l'altura de vol adient. En aquest cas, l'estalvi econòmic, mediambiental i de perillositat és important.

Una altra aplicació que pot ser de caràcter tant militar com civil és la vigilància en general. Per exemple, la frontera de Catalunya i Aragó amb França és força muntanyosa i, mitjançant una flota de UAVs equipada amb càmares es podria mantenir una vigilància intensiva amb una inversió clarament més petita que si es volgués realitzar amb aeronaus tripulades. Un altre cas de vigilància on els UAVs podrien fer un bon servei a la comunitat és la de carreteres secundàries de muntanya o com ja s'ha anomenat en la introducció en el servei d'alerta i rescat (SAR) de qualsevol país.

5.3. Desenvolupament futur

Un possible desenvolupament futur d'aquest treball seria la introducció de la component vertical a l'hora de salvar obstacles mitjançant, per exemple, diagrames de Voronoi 3D per

mantenir la filosofia de funcionament de l'eina. Aquest fet li donaria molt més dinamisme al programa, encara que també augmentaria la seva complexitat.

També hi ha possibilitats de desenvolupament futur en la generació dels diagrames. Existeix una extensió dels diagrames de Voronoi, anomenats diagrames d'Apoloni. Aquests consideren els obstacles com a circumferències i, per tant, podriem donar una veritable àrea als obstacles que generen els diagrames. El problema que tenen aquest diagrames és que les seves arestes ja no serien rectes, sinó que serien corbes. Si volguessim seguir amb la filosofia del programa, deixariem de necessitar l'algorisme de l'equació de la recta però necessitariem alguna mena d'algorisme de l'equació de la corba.

5.4. Impacte mediambiental

En quant a l'impacte mediambiental d'un hipotètic ús extensiu dels UAVs no pot ser més positiu. La inmensa majoria d'aeronaus civils no tripulades consumeixen menys que una de convencional. A més les tasques a les ques es poden destinar els UAVs poden ser beneficioses pel medi ambient; a més de la possible dedicació a tasques de prevenció d'incendis forestals existeixen d'altres com l'anàlisi de la contaminació de l'aire, entre daltres.

5.5. Valoració personal

Personalment, crec que ha estat interessant realitzar aquest projecte ja que per mi ha sigut una experiència molt enriquidora en diversos aspectes. El fet d'apendre a utilitzar un nou llenguatge de programació és molt positiu de cara a un futur. Saber les moltes possibilitats que té el Google Earth com a eina de representació geogràfica també ho trobo molt útil.

No obstant, amb el que més he disfrutat és amb la capacitat de resoldre els molts entrebancs que et van apareixen al llarg de qualsevol treball i, finalment, assolir els objectius proposats inicialment.

5.6. Agraïments

Només agrair al director d'aquest treball, Eduard Santamaria, pel temps dedicat.

BIBLIOGRAFIA

[1] Wikipedia. *Coordenadas Geográficas*.

http://es.wikipedia.org/wiki/Coordenadas_geogr\%C3\%A1ficas

[2] Wikipedia. *Coordenadas UTM*.

http://es.wikipedia.org/wiki/Coordenadas_UTM

[3] Salazar, Dagoberto: *La forma general de la Tierra*.

http://nacc.upc.es/tierra/node10.html

[4] GIS-SIG: Sistemas de Información Geográfica.

http//www.grabrielortiz.com/index.asp?Info=058a

[5] Institut Cartogràfic de Catalunya. *Model d'Elevacions de Catalunya-Revisio1*.

MET_Catalunya30_revisio1.doc

Wikipedia: *Modelo digital del terreno*.

http://es.wikipedia.org/wiki/Modelo_digital_del_terreno

Wikipedia: *Selva Negra*.

http://es.wikipedia.org/wiki/Selva_Negra

[6] Wikipedia: *Bresenham's line algorithm*.

http://en.wikipedia.org/wiki/Bresenham\%27s_line_algorithm

Medellín Anaya, Héctor E.:*Algoritmo basado en la ecuación de la recta*.

http://galia.fc.uaslp.mx/~medellin/Applets/LineasRectas/Recta.htm

[7] Wikipedia: *GDAL*

http://en.wikipedia.org/wiki/GDAL

GDAL: *GDAL - Geospatial Data Abstraction Library*

http://www.gdal.org/

[8] GDAL FAQ *What exactly was the license terms for GDAL?*

http://www.gdal.org/faq.html#license

[9] O'Rouke, Joseph. "Voronoi Diagrams". *Computational Geometry in C* Cambrige University Press. Cambrige. 1998 (Segona ed.): pàgs. 181-226.

de Berg, Mark et al. "Voronoi Diagrams". *Computational Geometry* Springer. Berlín. 2008 (Tercera ed.): pàgs 147-170.

Wikipedia: *Voronoi diagram*

http://en.wikipedia.org/wiki/Voronoi_diagram

Crew, Paul: *Voronoi/Delaunay applet*

http://www.cs.cornell.edu/Info/People/chew/Delaunay.html

Bendi: *CodeProject: Fortune's Voronoi algorithm implemented in Csharp*

http://www.codeproject.com/KB/recipes/fortunevoronoi.aspx

[10] Wikipedia: *GNU Lesser General Public License*

http://es.wikipedia.org/wiki/GNU_Lesser_General_Public_License

[11] Wikipedia: *Algoritmo de búsqueda*

http://es.wikipedia.org/wiki/Algoritmo_de_b%C3%BAsqueda

Chen, Dani Z. *Developing Algorithms and Software for Geometric Path Planning Problems*.

http://www.cs.brown.edu/people/rt/sdcr/chen/chen.html

[12] Wikipedia: *Teoría de grafos*

http://es.wikipedia.org/wiki/Teor%C3%ADa_de_grafos

[13] Wikipedia: *Wikilibro: C sharp NET* Capítols 0 i 1.

http://es.wikibooks.org/wiki/Programaci%C3%B3n:C_sharp_NET

[14] Wikipedia: *Google Earth*

http://es.wikipedia.org/wiki/Google_Earth

Google Earth: *Homepage*

http://earth.google.com/intl/es/