Artículo **Revista de Sistemas Computacionales y TIC's**
Diciembre 2017 Vol.3 No.10 50-57

Interfaz gráfica de planeación de la trayectoria de un vehículo aéreo no tripulado

GARCÍA-JIMÉNEZ, Omar*†, SERNA-ENCINAS, María, VALENCIA-PALOMO, Guillermo y ROSE-GÓMEZ César

Recibido Octubre 4, 2017; Aceptado Noviembre 7, 2017

Resumen

Originalmente los vehículos aéreos no tripulados (VANT), se utilizaban para reconocimiento militar o de combate. En la actualidad, es posible emplearlos con fines comerciales, civiles y de entretenimiento. Debido al gran potencial de estos vehículos y a su uso creciente, el presente trabajo se focaliza en desarrollar un sistema de planeación de trayectoria de vuelo, para extender la funcionalidad de un VANT, y así utilizarlos con fines médicos, de asistencia rápida, de transporte de carga, entre otros. La interfaz desarrollada presenta la capacidad del VANT de desplazarse de un punto origen a otro punto destino, determinando la ruta óptima entre éstos. Además, la interfaz genera un cuadrante con un enfoque de aproximación discreto y se utiliza el algoritmo A* para calcular trayectorias; se consideran los parámetros de menor distancia y menor número de giros para determinar dichas rutas. Como resultado, se tiene un sistema que permite visualizar un área en el mapa, extraer información del entorno para reconocer obstáculos, y convertir dicha área a un cuadrante a ser procesado, para determinar el cálculo de rutas óptimas. Finalmente, la interfaz presenta un conjunto de trayectorias óptimas de las cuales el usuario selecciona la ruta deseada.

Vehículo aéreo no tripulado (VANT), ruta óptima, algoritmo A*, planeación de trayectoria

Abstract

In its initial stage of development, the unmanned aerial vehicles (UAVs) were used for military recognition or military combat. Today, it is possible to use them for commercial, civil and entertainment purposes. Due to the great potential of these vehicles and their increasing use, the present work focuses on developing a flight path planning system to extend the functionality of a UAV, and to use them for medical purposes, fast assistance, and cargo transport, among others. The developed interface presents the ability of the UAV to move between two points, determining the optimal route between these points. Besides, the interface generates a quadrant to calculate trajectories using a discrete approach and the algorithm A*. The parameters of shortest distance and fewer turns are considered to determine these routes. Hence, the system allows to visualize an area on the map, extract information from the environment to recognize obstacles, and convert that area to a quadrant to be processed to determine the optimal route. Finally, the interface presents a set of optimal paths from which the user selects the desired route.

Unmanned aerial vehicles (UAVs), optimal route, algorithm A*, planning path

Citación: GARCÍA-JIMÉNEZ, Omar, SERNA-ENCINAS, María, VALENCIA-PALOMO, Guillermo y ROSE-GÓMEZ César. Interfaz gráfica de planeación de la trayectoria de un vehículo aéreo no tripulado. Revista de Sistemas Computacionales y TIC'S 2017, 3-10: 50-57

* Correspondencia al Autor (Correo Electrónico: omar.garciaj28@gmail.com)
† Investigador contribuyendo como primer autor.

Introducción

Una de las tecnologías más destacadas en estos últimos años, son los VANT, pequeños vehículos voladores no tripulados que pueden ser controlados de forma remota. Pueden usarse en infinidad de tareas que el humano no puede o no quiere realizar, ya sea por conveniencia personal, o porque dichas actividades resultan muy peligrosas, como la exploración en zonas de desastre, la limpieza de residuos tóxicos, la monitorización en zonas en guerra, entre otras. También han demostrado ser bastante útiles para el control de incendios forestales, operaciones geológicas, la agricultura, la construcción, entre otras. Una de las mayores fortalezas de este tipo de vehículos es lo económico de su operación, un consumo extremadamente bajo de combustible y piezas de recambio, además de salvaguardar la integridad física de los operadores de estos VANT.

Teniendo en cuenta las características de estos aparatos, se tiene como objetivo el brindar a un VANT la capacidad de autonomía de vuelo, para desplazarse desde un punto A hasta un punto B, determinando la ruta más eficiente. Para lograr esto, la investigación se focaliza en el desarrollo de un sistema que permita realizar la planeación de trayectoria de vuelo, y aprovechar las ventajas que ofrecen estos vehículos. La aplicación desarrollada consiste en la generación de un cuadrante con un enfoque de aproximación discreto, utilizando el algoritmo A* para la búsqueda y trazado de trayectorias. Se decidió utilizar este algoritmo debido a su eficiencia y a que su costo de almacenamiento en memoria exponencial, no representa un inconveniente teniendo en cuenta la plataforma destino. Para el cálculo de rutas se consideran los parámetros de menor distancia y menor número de giros. Como resultado, se tiene un sistema que permite visualizar un área en el mapa, extraer información del entorno para reconocer obstáculos, y convertir dicha área a un cuadrante a ser procesado, para determinar el cálculo de rutas óptimas.

Finalmente, la interfaz presenta un conjunto de trayectorias óptimas, de las cuales el usuario selecciona la ruta deseada.

Este artículo está compuesto de 4 secciones, en las cuales se describen diferentes aspectos tanto de la investigación, como del sistema en sí, además de información complementaria. En la segunda sección, titulada "Desarrollo", se muestra la metodología seguida para el desarrollo del proyecto y se describe el funcionamiento general de la aplicación, explicando el comportamiento de cada módulo, así como fragmentos de código de las funciones principales. En la tercera sección, titulada "Análisis de Resultados", se describe un procedimiento de prueba para comprobar el funcionamiento del sistema, así como los resultados obtenidos de la implementación del mismo. En la cuarta y última sección, se presentan las conclusiones obtenidas a partir de la implementación de la interfaz; y además, se incluye una perspectiva de trabajos a futuro.

Desarrollo

El sistema de planificación de trayectoria fue desarrollado en la plataforma JavaFX, derivada de Java, siendo ésta elegida debido a su capacidad para la creación de aplicaciones Web enriquecidas; en otras palabras, proyectos web que incluyen características propias de los programas de escritorio, permitiendo la incorporación de cualquier biblioteca de Java necesaria para su funcionamiento. El proyecto fue construido utilizando el entorno de desarrollo NetBeans, con el lenguaje FXML (derivado de XML) para la definición de las interfaces de usuario.

ISSN-2444-5002

En esta sección se presenta, primeramente, las fases que componen la metodología seguida a lo largo del proyecto de investigación; además, se describe a detalle tanto el análisis de la interfaz, como el diseño de la misma, y finalmente se muestra la arquitectura propuesta del sistema implementado.

Metodología

En la figura 1 se muestra la metodología seguida para la realización de la interfaz gráfica, compuesta de tres fases.

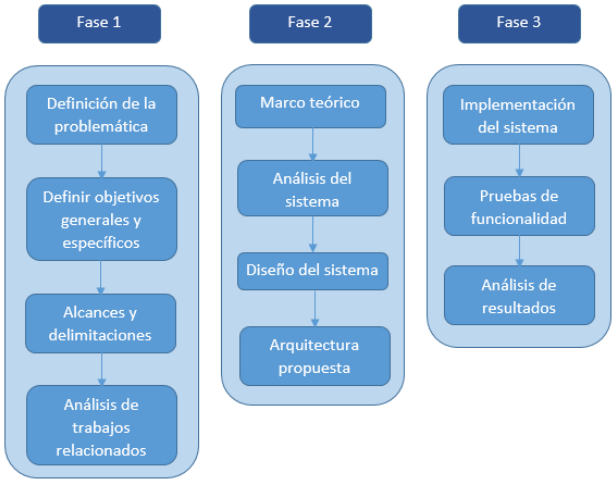

Figura 1 Metodología seguida para la realización del proyecto (Elaboración propia)

La primera fase de la metodología abarca la definición del problema a resolver, los objetivos planteados, los alcances y delimitaciones, y la última actividad consiste en el estudio de trabajos relacionados. La segunda fase consiste en el marco teórico que fundamenta el trabajo, describe a detalle el análisis y diseño de la interfaz y termina con el diagrama de la arquitectura propuesta del sistema. La tercera y última fase, incluye tanto la implementación del sistema como las pruebas de funcionalidad y el análisis de los resultados obtenidos.

Análisis de la interfaz

Como parte del análisis del proyecto, se derivan los diagramas representados en las figuras 2 y 3, que describen la interacción del usuario con los diferentes procesos que conforman la aplicación.

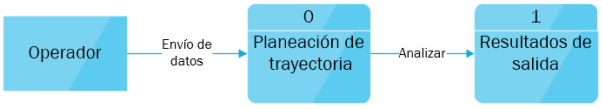

Figura 2 Diagrama de contexto de nivel 0

En la figura 2 se muestra el diagrama de contexto de la interfaz, el operador es la entidad que interactúa con los procesos de planeación de trayectoria y resultados de salida.

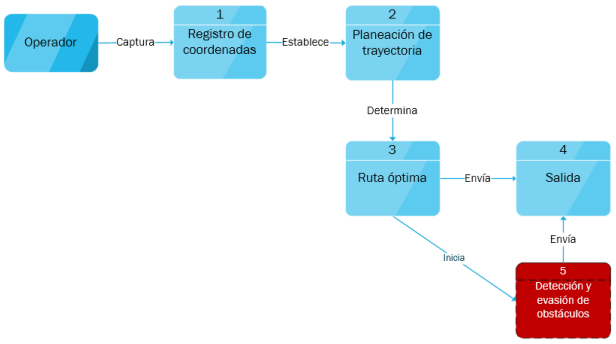

Figura 3 Diagrama de nivel superior: nivel 1

En la figura 3 se muestra a detalle la relación entre los procesos que conforman el sistema, el operador introduce las coordenadas del terreno seleccionándolo en el mapa, ubicando en él los puntos de origen y destino, éstos son utilizados por el algoritmo de planeación de trayectoria para el cálculo de rutas, y finalmente exportar los resultados obtenidos. A esto se le añade un proceso extra para la detección y evasión de obstáculos, en color rojo, como perspectiva de trabajo a seguir.

ISSN-2444-5002

Diseño de la interfaz

A continuación, se describe el diseño de las interfaces del sistema, así como las instrucciones necesarias para su construcción y funcionamiento interno.

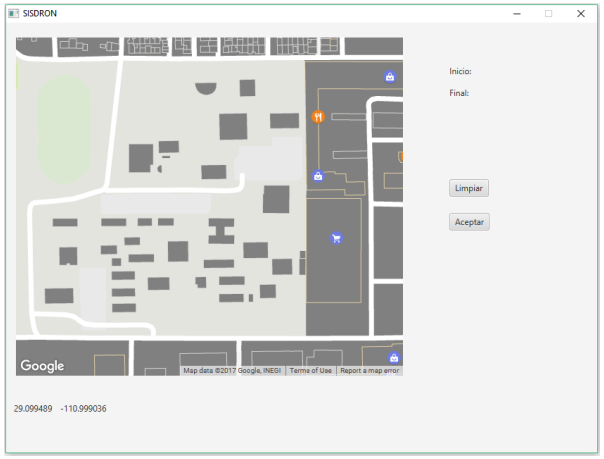

Figura 4 Interfaz inicial

En la figura 4 se muestra la primera pantalla del sistema, ésta contiene el mapa ubicado en la zona a trabajar, es en este punto donde se ingresan las coordenadas de origen y destino para el algoritmo de búsqueda de rutas.

```
public void start(Stage stage) throws Exception {
    URL location = getClass().getResource("/sisdron/Scene.fxml");
    FXMLLoader fxmlLoader = new FXMLLoader();
    Parent root = fxmlLoader.load(location.openStream());

    Scene scene = new Scene(root);

    stage.setTitle("SISDRON");
    stage.setScene(scene);
    stage.setResizable(false);
    stage.show();

    stage.addEventHandler(WindowEvent.WINDOW_HIDDEN, event -> {
        System.exit(0);
    });
}
```

Figura 5 Punto de entrada de la aplicación

La figura 5 muestra las instrucciones para construir la interfaz de inicio, cargando el archivo Scene el cual contiene el diseño de la vista en lenguaje XML, parcialmente representado en la siguiente figura.

```
<AnchorPane xmlns="http://javafx.com/javafx/8.0.65"
 xmlns:fx="http://javafx.com/fxml/1" id="AnchorPane" prefHeight="600.0"
 prefWidth="800.0" fx:controller="sisdron.SISDRONController">
   <children>
     <Label fx:id="label" layoutX="126" layoutY="120" minHeight="16"
     minWidth="69"/>
     <GoogleMapView fx:id="googleMapView" layoutX="14.0" layoutY="21.0"
     prefHeight="480.0" prefWidth="550.0"/>
     <GridPane layoutX="630.0" layoutY="20.0" prefHeight="95.0"
     prefWidth="200.0">...</GridPane>
     <GridPane layoutX="630.0" layoutY="200.0" prefHeight="95.0"
     prefWidth="200.0">...</GridPane>
     <GridPane layoutX="10.0" layoutY="500.0" prefHeight="95.0"
     prefWidth="200.0">...</GridPane>
   </children>
</AnchorPane>
```

Figura 6 Definición de la vista de la interfaz

En la figura 6 se resalta la instrucción que permite inicializar el objeto correspondiente al mapa, además de la carga del controlador asociado a ésta, el cual contiene la lógica que permite el funcionamiento de esta vista.

```
Scene scene = googleMapView.getScene();
WritableImage wI = scene.snapshot(null);

PixelReader reader = wI.getPixelReader();


Color color = reader.getColor(x, y);
if (!esPasable(color)) {
    nodoPasable = false;
}

private boolean esPasable(Color color) {
    int r = (int) (color.getRed() * 255);
    int g = (int) (color.getGreen() * 255);
    int b = (int) (color.getBlue() * 255);

    //Matriz de colores
    int[][] colores = {
        {228, 228, 223, 1},
        {128, 128, 128, 0},};
    for (int i = 0; i < 2; i++) {
        if (r == colores[i][0] && g == colores[i][1]
            && b == colores[i][2]){
            return colores[i][3] == 1;
        }
    }
    return true;
}
```

Figura 7 Controlador con instrucciones para reconocer obstáculos

Una vez ingresadas las coordenadas, lo siguiente es realizar el reconocimiento de obstáculos en la imagen del terreno. Como se muestra en la figura 7, aparecen las instrucciones en donde se ejecuta la parte principal de este proceso.

ISSN-2444-5002

Al tener capturada el área seleccionada en el mapa, la imagen se almacena en memoria, cuyos pixeles son analizados y reconocidos uno a uno por una función que determina si este pixel es pasable, para esto se extrae el color contenido en ese pixel, se descompone en sus valores RBG para compararlos con los existentes en la matriz definida en la misma función, y en caso de encontrarse una coincidencia, se determina si éste representa o no a un obstáculo.

Figura 8 Interfaz para búsqueda y exportación de rutas

Después de realizado el reconocimiento, se realiza la construcción de la estructura interna de la interfaz de búsqueda de rutas, como se trata de un cuadrante, éste se genera conforme a los resultados del reconocimiento, para posteriormente quedar graficado tal como se presenta en la figura 8. La estructura está construida de forma que pueda ser interpretada por el algoritmo A* implementado en la aplicación.

```
public void encontrarRuta() {
    int counter = 1;
    boolean fin = false;

    while (openSet.size() > 0 && !fin) {
        Nodo x = encontrarMenor(fScore);
        if (x.equals(goal)) {
            fin = true;
        }
        openSet.remove(x);
        fScore.remove(x.toString());
        closedSet.add(x);
        for (Nodo y : nodosCercanos(x)) {
            if (closedSet.contains(y)) {
                continue;
            }
            gScore.put(y.toString(), counter);
            counter++;
            int turn = 0;
            if (method.equals("turning"))
            {
                if(cambioDireccion(openDir.get(y)))
                {
                    turn = 1;
                }
            }
            int tentativeGScore = gScore.get(x.toString())
            + (int) distanciaEntre(x, y) + turn;
            openDir.remove(y);
            boolean tentativeIsBetter;
            if (!openSet.contains(y)) {
                openSet.add(y);
                tentativeIsBetter = true;
            } else if (tentativeGScore <= gScore.get(y.toString())) {
                tentativeIsBetter = true;
            } else {
                tentativeIsBetter = false;
            }
            if (tentativeIsBetter) {
                cameFrom.put(y.toString(), x);
                gScore.put(y.toString(), tentativeGScore);
                hScore.put(y.toString(), heuristicaEstimada(y, this.goal));
                fScore.put(y.toString(), gScore.get(y.toString())
                + hScore.get(y.toString()));
                dir = openDir.get(y);
            }
        }
    }
}
```

Figura 9 Instrucciones principales para la ejecución del algoritmo A*

Para la ejecución del algoritmo de búsqueda se utiliza el cuadrante generado en el paso anterior. Se hace un recorrido de cada uno de los nodos del cuadrante para determinar cuáles son los transitables, éstos son guardados en una lista, la cual es introducida dentro de la función representada en la figura 9, y ésta devuelve las rutas resultantes, a cada una de ellas les es asignado un color aleatorio, para ser reconocidas fácilmente en el mapa, el algoritmo devuelve rutas eficientes en términos de menor distancia, o menor número de giros.

```
private Boolean cambioDireccion(String dirA)
{
    if (!dir.isEmpty())
    {
        if (!dir.equals(dirA))
        {
            return true;
        }
    }
    return false;
}
```

Figura 10 Modificación del algoritmo A* para priorizar la reducción del número de giros

Debido a que, para los vehículos aéreos, los cambios de dirección representan un costo adicional, es recomendable reducirlos lo más posible, por lo que al algoritmo A* se le realizó una modificación para considerar este caso particular. Cuando se realiza un desplazamiento a la próxima celda, el sistema guarda en memoria la dirección hacia la que se movió, entonces la función representada en la figura 10 consulta la dirección guardada en memoria, comparándola con la nueva dirección hacia la que se desplazó la celda actual, y si estos valores difieren, se determina que es un cambio de dirección, y se añade el peso extra para que el algoritmo reste prioridad a esa ruta en caso de contener demasiados giros.

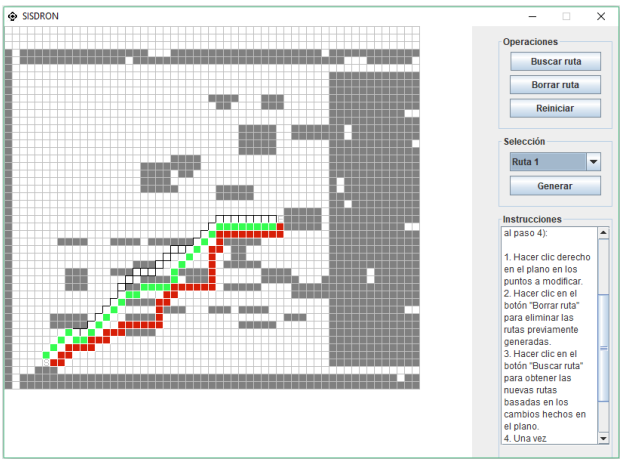

Figura 11 Interfaz con las rutas generadas

Al finalizar la ejecución del algoritmo, en el mapa aparecen las rutas encontradas, cargadas en el menú de selección, y dependiendo de la ruta seleccionada, ésta es resaltada en la interfaz mediante un parpadeo que alterna entre el color asignado y el blanco. Al elegir una ruta y seleccionar la opción "Generar", se exportará la información correspondiente de dicha ruta a un archivo, para su posterior uso.

Arquitectura del sistema

En la figura 12 se representa la arquitectura propuesta para el sistema, ésta consta de tres módulos: Registro, Planeación de trayectoria y Resultados de salida.

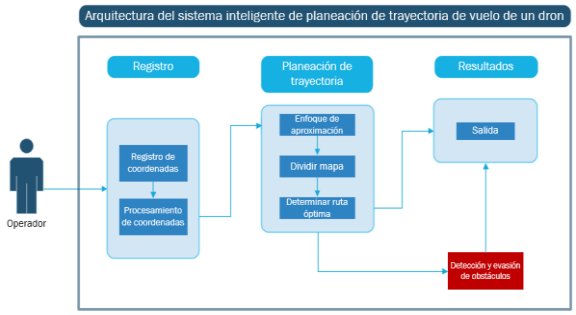

Figura 12 Arquitectura propuesta

El primer módulo muestra el mapa donde se introducen las coordenadas de los puntos de origen y destino a recorrer, para posteriormente ser procesados; en el segundo módulo se realiza el tratamiento de la información obtenida del módulo de registro, para el cálculo de rutas correspondientes al terreno seleccionado; y el tercer módulo es el que registra los datos resultantes del trayecto, seleccionado entre aquellos que fueron encontrados y calculados en el proceso anterior.

Resultados obtenidos

El resultado final de todos los procesos antes descritos, es el de un archivo exportable con las coordenadas que conforman el trayecto elegido para que sea recorrido por el VANT.

ISSN-2444-5002

Para lograr esto, se obtienen los datos generados por el algoritmo de búsqueda, y se concentran en una estructura para formar el archivo antes mencionado.

Name	Type	Value
[0 - 99]		
[0]	Nodo	#3695
x	int	0
y	int	0
ancho	int	10
alto	int	10
pasable	boolean	true
inicio	boolean	false
_final	boolean	false
[1]	Nodo	#3696
[2]	Nodo	#3697
[3]	Nodo	#3698
[4]	Nodo	#3699
[5]	Nodo	#3700

Figura 13 Parte de los resultados al momento de ejecutar el algoritmo de búsqueda

En la figura 13 se muestra una sección de los resultados obtenidos mediante el algoritmo A*, mien tras realiza el cálculo de rutas, organizándolos en una estructura basada en nodos, y almacenándolos para su posterior exportación. Puede apreciarse la estructura de un nodo individual que resultó ser un nodo libre para el tránsito por esa sección del terreno, es decir, sin obstáculos presentes.

```
29.101158893402058 -110.99905642991638
29.101065146819195 -110.99905642991638
29.100971400236332 -110.99905642991638
29.100877653653473 -110.99905642991638
29.10078390707061 -110.99905642991638
29.100690160487748 -110.99905642991638
29.100596413904885 -110.99905642991638
29.100502667322022 -110.99905642991638
29.100408920739163 -110.99905642991638
29.1003151741563 -110.99905642991638
29.100221427573437 -110.99905642991638
29.100127680990575 -110.99905642991638
29.100033934407712 -110.99905642991638
29.09994018782485 -110.99905642991638
29.09984644124199 -110.99905642991638
29.099752694659127 -110.99905642991638
29.099658948076264 -110.99905642991638
29.0995652014934 -110.99905642991638
29.09947145491054 -110.99905642991638
29.09937770832768 -110.99905642991638
29.099283961744817 -110.99905642991638
29.099190215161954 -110.99905642991638
29.09909646857909 -110.99905642991638
29.09900272199623 -110.99905642991638
```

Figura 14 Archivo de texto con las coordenadas

En la figura 14 se muestra un archivo de texto plano, éste contiene las coordenadas latitud-longitud, correspondientes a los nodos que conforman la ruta elegida como resultado de la búsqueda hecha por el sistema. Este archivo puede utilizarse en cualquier VANT compatible con la funcionalidad de interpretación de estas coordenadas.

Conclusiones

A partir de la información presentada en este artículo, puede concluirse que el sistema desarrollado permite realizar la planeación de trayectoria de un VANT, definiendo los puntos de origen y destino para determinar la ruta óptima entre ambos. Al visualizar un área en el mapa, se reconocen los obstáculos presentes, convirtiendo en cuadrante el terreno en cuestión, para sobre éste realizar la búsqueda de rutas, generando un archivo con las coordenadas correspondientes a ésta.

ISSN-2444-5002

Este proyecto será útil para aumentar las capacidades de un VANT, consiguiendo una mejoría en el desempeño de este tipo de vehículos en sus recorridos.

El sistema puede refinarse y extenderse, añadiendo más opciones de configuración e ingreso de parámetros para obtener resultados más especializados, así como para hacer modificaciones en el reconocimiento de obstáculos, en caso de ser necesario.

Además, se tiene proyectado añadir la función de detección y evasión de obstáculos, tomando como base la investigación realizada en este trabajo acerca de sensores, su funcionamiento entre otros temas, para así desarrollar esta funcionalidad e integrarla a este sistema.

Agradecimiento

Se agradece al Consejo Nacional de Ciencia y Tecnología la contribución para el desarrollo de esta investigación mediante la beca 708775/585611, otorgada al primer autor

Referencias

B. Martínez-Jiménez, L. Pineda-Bombino, M. Martínez-Carmenate, D. De-Ávila-Rodríguez, & L. Hernández-Santana. (2012). Identificación de un vehículo aéreo no tripulado. Ingeniería Electrónica, Automática y Comunicaciones, 33(1), 45-55.

N. Correl. (2011). Introduction to Robotics #4: Path-Planning. [En línea]. Disponible: http://correll.cs.colorado.edu/?p=965 (Visitado el 15 de marzo del 2016)

M. Nieuwenhuisen, D. Droeschel, M. Beul & S. Behnke. (2014). Obstacle Detection and Navigation Planning for Autonomous Micro Aerial Vehicles. University of Bonn.

B. Holman. (2009). The first air bomb: Venice, 15 July 1849. [En línea]. Disponible: https://airminded.org/2009/08/22/the-first-air-bomb-venice-15-july-1849/ (Visitado el 02 de septiembre del 2016).

G. Addati, & G. Lance. (2014). Introducción a los UAVs, Drones o VANTs de uso Civil (No. 551). Universidad Del CEMA.