-Todos los elementos necesarios junto con el código fuente implementado.

-Una memoria en la que se indiquen los nombres de los alumnos.

-Una explicación detallada de todos los elementos programados.

-Todos aquellos aspectos que se consideren necesarios para probar correctamente el funcionamiento de la práctica.

-Dicha memoria también incluirá un apartado en el se reflejen los resultados obtenidos durante la experimentación (por ejemplo, gráficas con los datos relativos a la evolución del fitness en el proceso).

-Un apartado de conclusiones y posibles mejoras/ampliaciones que se hayan detectado durante el desarrollo de la práctica.

Introduccion, -Drone –Controlador borroso

Explicacion del código -Drone –Controlador borroso

Resultados obtenidos

Conclusiones

Mejoras

Anexo Como Ejecutarlo

El sistema que se va a presentar es un controlador borroso el cual evitará obstáculos en el plano horizontal a la par que se dirige hacia el punto objetivo. Las entradas de nuestro sistema son: distancia al obstáculo, ángulo del drone con respecto al obstáculo y ángulo del drone con respecto al objetivo. Como salida del controlador borroso tendremos el ángulo que deberá girar el drone.

Lo malo de probarlo nuestro sistema en un drone es que tiene inercia y por lo tanto cuando tiene que girar en sentido contrario para corregir la dirección tarda en hacerlo debido a las fuerzas.

Este sistema es utilizable para cualquier vehículo, incluso para terrestres.

**Explicacion del código**

Codigo fuente en Java

En este lenguaje se ha utilizado para programar todo el código correspondiente al controlador borroso. Esto es debido a que se ha empleado la librería jFuzzyLogic, la cual usa el lenguaje Java para implementar el lenguaje de control Fuzzy.

Dado que Unity no permite el lenguaje Java se ha creado un cliente en Unity que conecta con un servidor en Java. En primer lugar esta conexión se realizó mediante el protocolo TCP pero tras realizar algunas pruebas de rendimiento se optó por usar el protocolo UDP ya que tanto cliente como servidor se ejecutan en local. Mientras que TCP bajaba el rendimiento a 17-22 frames por segundo, UDP era capaz de doblar el rendimiento de TCP, lo cual mejoraba el rendimiento grafico del sistema.

Clases Java para implementar el controlador borroso

* obstacleAvoidance.java

Clase encargada de gestionar el controlador borroso. Gracias a su constructor se carga el fichero fcl con el cual se crea un objeto “Fuzzy Inference System”. También tiene un método nombrado “evaluar” el cual se encarga de evaluar las entradas del controlador borroso (distancia al obstáculo, ángulo al obstáculo y ángulo objetivo) para dar una salida (ángulo de giro del drone).

Fichero FCL

* Este fichero define el sistema de inferencia borroso (Fuzzy Inference System).

Clases Java empleadas para realizar pruebas

* servidorEchoTcp.java

Primera clase implementada para realizar la conexión con Unity. Se trata de un servidor el cual recibe un mensaje y lo envía de vuelta al cliente. Con esta clase se comprobó el rendimiento del protocolo TCP y se decidió descartar el uso del mismo.

* servidorFuzzyComentado.java

Es una copia de la clase servidorFuzzy.java en la cual se añaden comentarios para poder seguir y depurar lo recibido y enviado por el servidor.

* obstacleAvoidanceComentado.java

Es una copia de la clase obstacleAvoidance.java esta clase implementa el controlador borroso gracias al fichero fcl. Pero a diferencia de la clase sin comentarios, en esta podremos ver las variables evaluadas, un gráfico con el valor que toman las variables de entrada en el rango de cada una de ellas, un gráfico con el valor de salida del controlador y todas las reglas que se han activado y con qué peso dadas las variables de entrada.

* pruebaEvaluar.java

Gracias a esta clase y a la clase anterior pudimos evaluar caso a caso que reglas se activaban dadas unas ciertas variables de entrada (muy útil cuando se detectan comportamientos anómalos en la ejecución). Gracias a lo cual pudimos evaluar la configuración del fichero obstacleAvoidance.fcl y adaptarlo a un comportamiento más realista.

¿Como probar la última versión?

En primer lugar se utilizará al aplicación de escritorio Eclipse para abrir todos los archivos “.java”. Para ello se abrirá Eclipse y se seleccionará como directorio la siguiente ruta dentro de la carpeta descargada: “JAVA\workspace”. Una vez hecho esto, en el menú superior de eclipse se seleccionará File>Import>Existing Projects into Workspace. Se abrirá una ventana en la cual seleccionaremos la ruta de nuestro directorio clickando en “browse”, donde buscaremos la ruta de la carpeta descargada y dentro de ella seleccionaremos JAVA\workspace\Jfuzzy. Con esto ya tenemos nuestro proyecto importado.

En el caso de que en package explorer (a la izquierda de la pantalla) nos salga nuestro proyecto con un aspa roja, daremos click derecho al proyecto Jfuzzy y seleccionaremos properties. En la ventana que se abre seleccionaremos “java build path”, después “libraries”. En este punto borraremos jFuzzyLogic.jar seleccionándolo y dándole a remove. Despues añadiremos nuestro JAR dando al botón “add external jars” y seleccionaremos el JAR el cual está en la carpeta descargada en la ruta “JAVA\workspace\Jfuzzy.”

El siguiente paso es desplegar nuestro proyecto y meternos en el paquete fuzzy, después seleccionaremos la clase servidorFuzzy y daremos al icono run (verde con un “play” blanco en su interior) para ejecutar la clase. (en caso de querer ver que se envía y se recibe en el servidor se recomienda ejecutar la clase servidorFuzzyComentado del paquete “pruebas\_fuzzy” en lugar de la clase servidorFuzzy.

En este punto ya tenemos arrancado el servidor (puerto 9900, protocolo UDP).

En segundo lugar tenemos que abrir el proyecto de Unity. Para ello se abrirá la aplicación de escritorio Unity 5.4.1f1 y se abrirá el proyecto seleccionando la carpeta descargada. Una vez abierto, se seleccionará la escena “\_Scene” y se pulsará el botón play.

-- dejar comentarios bonitos.

--dejar las clases y los scripts bonitos.

--Dejar las cosas comentadas y vigilar que todo este con nombre buenos.

Fran ya lo ha hecho ;D

**INTRODUCCIÓN**

El desarrollo de esta práctica tiene como objetivo el desarrollo de un drone autónomo. Dado un punto de destino el drone debe de ser capaz de alcanzar dicho punto evitando los obstáculos presentes entre el dron y el destino.

El entorno del drone se desarrolla en Unity. Por tanto, hacemos uso de las físicas proporcionadas por éste para conseguir un resultado lo más parecido a la realidad.

**DRONE**

El drone es un cuadricóptero y tiene las siguientes características.

En primer lugar, el drone tiene asignada una masa y dispone de cuatro motores que aplican fuerzas independientes para simular el comportamiento realista del drone. Dependiendo de las fuerzas aplicadas a cada motor el dron realizará un movimiento diferente.

Debido a que se hace una simulación realista del funcionamiento del drone fue necesario implementar un control de estabilización para que el drone fuese capaz de mantenerse en el aire a una determinada altura. Para el desarrollo de este sistema de estabilización se implementó un PID para los cuatro motores que es capaz de mantener estable el drone cuando está en idle. Además, también se ha implementado un PID para cada movimiento del drone para que estos movimientos sean constantes. Los distintos tipos de movimientos del drone son los siguientes:

* Throttle: movimiento del drone que le permite aumentar o disminuir la altura.
* Yaw: movimiento del drone que le permite rotar sobre sí mismo.
* Pitch: movimiento del drone que le permite inclinarse hacia delante y hacia detrás
* Roll: movimiento del drone que le permite inclinarse hacia la izquierda o la derecha.

La representación de la estructura del drone es la siguiente.

INCLUIR IMAGEN CON EL DRONE, SUS MOTORES Y TIPOS DE MOV

**CONTROLADOR BORROSO**

El desarrollo de la inteligencia artificial se hace a través de un controlador borroso implementado en JFuzzy.

Para el desarrollo de este sistema se ha establecido que el drone sólo avance hacia delante y sólo sea capaz de girar respecto de su propio eje, es decir, no puede hacer uso de las funciones de inclinación hacia los lados.

El desarrollo del sistema basado en reglas fuzzy se puede separar en dos partes.

En primer lugar, la capacidad de evitar obstáculos que está basada en la recreación de una fuerza repulsiva del drone respecto de los obstáculos. Para la creación de esta parte se toman como entradas la distancia del obstáculo más cercano y el ángulo al que se encuentra el obstáculo respecto del drone. Para la creación de este sistema de detección de obstáculos el drone cuenta con una esfera de un cierto radio que lo recubre.

En segundo lugar, se recrea una fuerza de atracción del drone hacia la posición destino. Para su realización tomamos como entrada el ángulo respecto de la dirección del drone y la línea recta entre la posición actual del drone y la posición del destino.

Con todas estas entradas generamos una salida que le dice al drone cuanto debe girar para cumplir sus objetivos.

La representación del sistema del controlador borroso es la siguiente.

IMAGEN ENSEÑANDO LAS ENTRADAS DEL DRONE

**EXPLICACION DEL CODIGO**

**DRONE**

En este apartado se va a explicar la función de cada script necesario para el funcionamiento del drone.

*Controller.cs*

Este script se diseñó inicialmente para el control del drone usando el teclado. Posteriormente se adaptó para su uso con la inteligencia artificial.

*BasicControl.cs*

Este script se encarga de actualizar la velocidad de cada motor del drone dependiendo del movimiento que esté realizando.

*ComputerModule.cs*

Este script se encarga de actualizar la velocidad de un motor del drone.

*BasicGyro.cs*

Este script implementa un giroscopio simple para actualizar la rotación del drone dependiendo del movimiento que esté realizando.

*PID.cs*

Implementación de un PID.

*POV.cs*

Se encarga de mantener la cámara encima del drone.

*Motor.cs*

Se encarga de aplicar las fuerzas a cada motor.

*Move.cs*

Se encarga de mantener los ejes X y Z sin ninguna rotación cuando el drone se inclina. Con esto evitamos que las entradas del controlador fuzzy sean las correctas y no tengan ningún tipo de rotación aplicada.

*Fitness.cs*

Este script se encarga de comprobar cuando el drone ha alcanzado el destino y finaliza en tal caso. Además, también lleva la cuenta del número de colisiones sufridas por el drone.

*Sensing.cs*

Este script se encarga de obtener todas las entradas necesarias para el controlador fuzzy. En primer lugar, creamos nuestra esfera que recubre el drone y se encarga detectar los obstáculos dentro de su volumen y obtenemos la distancia a la que se encuentran y su ángulo respecto del drone. También obtenemos la distancia a la que está el destino y su ángulo respecto el dron.

**CONCLUSIONES**

Podemos concluir que el desarrollo de un drone autónomo ha sido exitoso. El drone es capaz de evitar obstáculos y llegar al destino de manera autónoma.

**MEJORAS**

Las posibles mejorar al drone son las siguientes.

* Sistema de control de inercia: en el diseño actual el drone no es capaz de manejar la inercia por lo que sería una buena mejora para evitar las colisiones producidas por ésta.
* Mejora del controlador borroso: se podría mejorar añadiendo nuevas entradas para que el dron fuese capaz de evaluar varios obstáculos a la vez.
* Finalización del controlador evolutivo: terminar la implementación a medio hacer en el diseño actual para que el sistema sea capaz de adaptar el peso de las reglas automáticamente.