



Escuela Técnica Superior de Ingenieros Informáticos

Universidad Politécnica de Madrid

Análisis de VREP como herramienta de simulación para Aerostack

Trabajo Fin de Máster

Máster Universitario en Inteligencia Artificial

AUTOR: Pablo Parodi Félix

TUTOR: Martín Molina González

Agradecimientos

*A Martín, por su paciencia y orientación;
y a mi familia y amigos que, con su compañía,
me allanan siempre el camino hacia cualquier objetivo.*

Resumen

La robótica está asumiendo un papel cada vez más importante en el mundo, especialmente en la industria, donde cada día aumenta el número de empresas que hacen uso de robots para automatizar tareas de producción, logística o inspección, entre muchas otras. Tradicionalmente, estos han llevado a cabo sus funciones en espacios de trabajo cercados, de acceso restringido a los trabajadores u otros robots por cuestiones de seguridad. No están diseñados para lidiar con el imprevisible y numeroso conjunto de variables en el que se podría ver envuelto en un entorno tan dinámico como es el mundo real, repleto de factores externos tan difíciles de modelar a priori como las trayectorias de estos o los obstáculos inesperados en el caso de robots móviles.

Gracias al creciente avance en las capacidades computacionales, se ha conseguido evolucionar hacia modelos de robots cada vez más autónomos que realizan las tareas para las que fueron diseñados bajo una nula o mínima supervisión, siguiendo la planificación propia del robot acorde, no sólo al análisis previo a su realización sino también al continuo procesamiento de información que va obteniendo en tiempo real a través de los sensores.

A medida que crece la complejidad en la robótica, se hace crucial la fase previa a la ejecución y puesta en marcha en el mundo real para probarlo antes y así evitar costes innecesarios. Los nuevos avances en el campo de la robótica hacen del testing un reto, pues requiere un riguroso criterio para su elaboración, de tal manera que ésta

cubra todos los movimientos, tareas o escenarios, que permita llevar a cabo un análisis cuantitativo con resultados medibles del desempeño del robot.

El presente documento pretende abordar y describir todos los elementos que conforman la robótica aérea, en general, y el testing, en particular, para así demostrar por qué llega a ser un campo en la misma línea de importancia que el resto de fases de desarrollo de un robot. Asimismo, se realiza un estudio de viabilidad de la herramienta VREP como simulador para Aerostack, la herramienta software desarrollada por el grupo de investigación de la Universidad Politécnica de Madrid Vision4UAV. Finalmente, se lleva a cabo un caso práctico concreto en el cual se analizan las necesidades de evaluación que requiere, se diseñan varios entornos de simulación.

Abstract

Robotics is taking on an increasingly relevant role in the world especially in industry where the number of companies using robots to automatize production, logistic and inspection tasks, among others, increases every day. Traditionally, they have carried out its functions in fenced workspaces of restricted access to workers or other robots for security reasons. They are not designed to deal with such an unpredictable and large set of variables that might be involved, in such a dynamic environment as real world is, with plenty of external factors so hard to model a priori, like its trajectories or the unexpected obstacles in case of mobile robots.

Thanks to increasing advance in the computational capabilities, developing towards robots models more and more autonomous that carry out the tasks they were designed for under no or minimal supervision is becoming possible. They can now follow the very robot plan according, not only to the previous analysis to the realization but also to the continuous processing of information that obtains in a real-time manner from the sensors.

As complexity in robotics grows, previous phases to the execution and start-up in real world becomes more important due to costs. New advances in robotic field make a challenge out of the testing, since it requires a rigorous criterion to be elaborated, so that it tackles every possible movement, task or scenes and it lets carry out a quantitative analysis with measurable results about robot performance.

It is herein aimed to address and describe all the elements that forms aerial robotics in general and the testing within it in particular, thus proving why it gets to

be a field so important as the rest of the robot development phases. Moreover, it is carried out a feasibility study of VREP software as simulator tool for Aerostack, a software tool developed by a researching group called Vision4UAV of Universidad Politécnica de Madrid. Finally, some virtual simulation environments are fully designed where the assessment needs it requires are previously analyzed.

Introducción	1
Objetivos	2
Estructura	3
Fundamentos Tecnológicos	4
Robótica aérea	5
Tipo de Drones	6
Metodología para la Inteligencia en Robots Autónomos	6
Sistemas Basados en el comportamiento	6
Aprendizaje	6
Adquisición de datos y sensorización	6
Sensores de proximidad, contacto y fuerza	6
Sensores inerciales	6
Sensores de rango	6
Descripción del Problema	6
Importancia de validación	6
Metodología de Evaluación	6
Software empleado	6
Aerostack	6
VREP	6
Aspectos a evaluar	6
Desarrollo del Método	6
Descripción de escenarios	6
Escenario 1. Persecución	6
Escenario 2. Inspección	6
Escenario 3. Inventario	6
Resultados	6
Análisis Grado de Adecuación	6
Conclusiones	6

INDICE DE FIGURAS

<i>Figura 1. Dron flapping-wing de combustión interna</i>	<i>2</i>
<i>Figura 2. Representación de un posible sistema de 3 capas.....</i>	<i>2</i>
<i>Figura 3. Representación en grafo de un sistema de aprendizaje basado en redes.....</i>	<i>2</i>
<i>Figura 4. Posible árbol generado por aprendizaje</i>	<i>2</i>
<i>Figura 5. Comparativa gráfica de sensores de rango.....</i>	<i>2</i>
<i>Figura 6. Esquematización de la arquitectura de control de Aerostack.....</i>	<i>2</i>
<i>Figura 7. GUI de VREP. Ventana principal</i>	<i>2</i>
<i>Figura 8. Escenario para persecución de drones.....</i>	<i>2</i>
<i>Figura 9. Escenario para simulaciones de tareas de inventario.....</i>	<i>2</i>
<i>Figura 10. Salida de comando 'top' en consola Ubuntu</i>	<i>2</i>
<i>Figura 11. Utilización de recursos para distinto número de robots</i>	<i>2</i>
<i>Figura 12. Utilización de recursos para distintas velocidades de simulación</i>	<i>2</i>
<i>Figura 13. Resultados prueba tiempo para 4, 8 y 16 drones</i>	<i>2</i>

INTRODUCCIÓN

El testing o validación es un proceso que consiste en la búsqueda de las condiciones límite que podrían conducir un sistema a fallo. Requiere un minucioso análisis en el que se exploren todos los posibles disturbios, entradas al sistema, o escenarios en los cuales el objeto a validar se tope con las adversidades que, de manera predefinida, debería estar capacitado para superar.

Este análisis se convierte en una fase más en el desarrollo de sistemas robóticos, que va creciendo en complejidad y exigencia a medida que se le brinda más autonomía e inteligencia al sistema. Con la creciente popularidad y continua evolución de campos como la inteligencia artificial, internet de las cosas o minería de datos, se abre la puerta a una nueva era de robots, con capacidades de procesamiento de datos obtenidos de su entorno en tiempo real para toma de decisiones difícilmente preprogramables. De esta manera, un robot puede llevar una carga de un lugar a otro, siguiendo una trayectoria ya planificada pero sujeta a posibles modificaciones, que deben obtenerse en el menor tiempo posible y con muy poca capacidad de memoria, realizadas para evitar accidentes, por ejemplo.

Estos avances, primeramente, dan paso a los robots colaborativos, los cuales ya prescinden de un espacio de trabajo único limitado por motivos de seguridad, y además abre un amplio abanico de aplicaciones posibles, civiles, industriales y militares, desde inspecciones industriales a reparto de mensajería, pasando por cinematografía, ensamblaje colaborativo, detección y respuesta a desastres naturales y un largo etcétera.

Para todas estas aplicaciones en las que difícilmente se puede modelar el sistema teniendo en cuenta la infinidad de posibles factores externos, imprevisibles o

arbitrarios algunos, es de suma importancia la validación previa del sistema en un entorno de simulación en el que se pueda poner a prueba bajo estos supuestos.

Las fases que conforman el diseño íntegro de un robot comprenden un complejo sistema de variables —cinemáticas, dinámicas, de control, etc.—, cuyo modelado puede ser costoso y, sobre todo, contener errores o conducir a fallos difícilmente previsibles. Poner este diseño en marcha hasta su construcción y puesta en ejecución en el mundo real supone un importante coste en tiempo y dinero; y exponerse al riesgo de que ocurra un fallo que dañe o rompa el mismo u otros robots, mobiliario, o incluso personas lo convierte en algo peligroso e inviable económicamente.

A partir de todo esto, nace la necesidad de validar mediante simulación como solución de bajo coste para el testeo de robots. Mediante el diseño de un modelo virtual del robot, llamado también modelo de simulación, es posible verificar el correcto movimiento del robot, así como su comportamiento dinámico y el de las leyes de controles bajo las cuales actúa. Para validar las hipótesis realizadas durante su diseño, el modelo de simulación debe reproducir fielmente a aquel que fuese a tener en el mundo real. Sin embargo, y como se ha mencionado anteriormente, no es una tarea trivial modelar todo el sistema teniendo en cuenta todas y cada una de las posibles interacciones que pueda tener éste con el entorno.

1.1. Objetivos

Comentado lo anterior y para algunas de las misiones, actualmente en fase de desarrollo, del grupo Computer Vision and Aerial Robotics (CVAR) de la Universidad Politécnica de Madrid, el objetivo principal del presente proyecto es el estudio del grado de adecuación de VREP como simulador y el diseño de un entorno de simulación para pruebas, o test-site simulado, mediante la herramienta V-REP y la elaboración del procedimiento a seguir para su validación. Para llevar a cabo esto, ha sido necesario alcanzar una serie de metas subsecuentes que se enumeran a continuación:

- Estudio bibliográfico sobre el estado del arte en las misiones y tareas más avanzadas en el campo los robots autónomos. Introducción a los distintos UAS, sus elementos de sensorización, técnicas de adquisición y tratamiento de datos y métodos de aprendizaje.
- Análisis de las limitaciones, necesidades y dificultades de los UAVs, especialmente en espacios cerrados e interiores, donde la geolocalización no es la opción más precisa.
- Aprendizaje y uso en profundidad del uso de la herramienta software de simulación V-REP, así como de la herramienta Aerostack-4UAVs.
- Análisis del problema a evaluar para obtener estimadores y variables de scoring para conocer el grado de adecuación.
- Diseño de modelos virtuales de simulación para pruebas o test-site que contengan todos los elementos necesarios para realizar las pruebas.

1.2. Estructura

La presente memoria se divide en distintos capítulos que recogen, de manera explícita y ordenada, todos los pasos seguidos durante el desarrollo del proyecto.

A partir de esta sección, el presente documento comienza, bajo el título de fundamentos tecnológicos, introduciendo al lector en el campo de la robótica, prestando especial atención en los robots aéreos. Se describen los tipos de robots voladores, así como la lógica que sigue la clasificación de estos en función de sus capacidades, como manejabilidad o autonomía de vuelo; se mencionan las principales aplicaciones de este tipo de robots y se pasa a describir los más importantes métodos y técnicas que consiguen proveerlos de un comportamiento autónomo e inteligente.

Continúa esta sección, centrándose ahora en los componentes usados para sensorización y adquisición de datos.

Con el fin de dar una visión sobre cómo son las herramientas software existentes en el mundo de la programación de estos tipos de robots autónomos, se describe de manera general la herramienta software Aerostack-4UAVS desarrollada por el grupo de investigación *Vision4UAV* de la *UPM*, así como *VREP*, que será la herramienta mayormente usada.

Una vez introducidos en materia, el documento se focaliza en justificar las necesidades de validación de operaciones o misiones, describir el tipo de misiones que podrían validarse, y el diseño y la construcción del entorno virtual de simulación test-site en *V-REP* en el capítulo cuatro. Se termina con un capítulo a modo de conclusiones, donde se analizan los resultados, se comparten impresiones y se discute sobre las necesidades o limitaciones descubiertas.

FUNDAMENTOS TECNOLÓGICOS

Teniendo en cuenta los requisitos establecidos en los objetivos del proyecto, se hace necesario un repaso de las principales ramas de la robótica aérea, sus características y aplicaciones, así como los elementos que lo componen y los métodos seguidos para el desarrollo de los sistemas que determinan su comportamiento durante la ejecución de tareas con un mayor nivel de autonomía. Así, de esta forma se introduce al lector en el campo del *testing*, argumentando las necesidades de su realización. En este apartado, se introducirá al lector en los drones y se describirán las principales tecnologías y enfoques metodológicos usados para el desarrollo de UAS autónomos, como las jerarquías de comportamiento, los métodos de aprendizaje, el procesamiento y tratamiento de los datos recogidos por los sensores, etc.

2.1 Robótica aérea

La robótica aérea abarca todo el conjunto de vehículos voladores que no son manejados o pilotados directamente por humanos. El término habitualmente usado para referirse a estos robots, UAV, viene de las siglas de Unmanned Aerial Vehicles. UAS se refiere al sistema completo formado por el UAV y demás elementos necesarios como pueden ser un IHM (interfaz humano-máquina), comunicaciones, etc. También se les conoce comúnmente como drones y será la palabra que se utilizará a partir de ahora para referirse a ellos.

Comparado con los clásicos robots móviles (terrestres) o brazos manipuladores, los ingenieros deben enfrentarse a retos mayores durante el diseño

de un dron debido a la complejidad que supone su condición voladora que dificulta el sistema en términos de propulsión, control y navegación, entre otros. Si bien es cierto que se han desarrollado sistemas híbridos que alargan y mantienen la batería de un dron mediante el uso de motores de combustión^[11], la primera limitación que se encuentra al analizar diferencias aparece en el consumo y por tanto en la autonomía energética, ya que, por causas derivadas de escala, de relación tamaño-peso, no es viable el uso de motores de combustión para misiones o tareas que requieran un largo tiempo o recorrido.

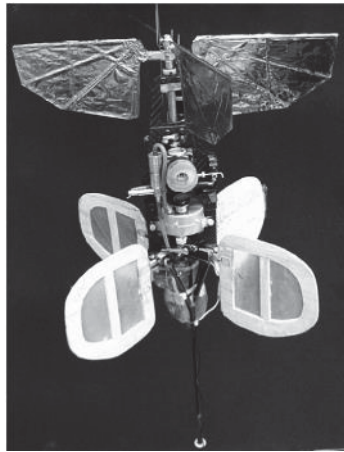


Figura 1. Dron flapping-wing de combustión interna

Por otra parte, se encuentra el riesgo de agotar la batería, pues mientras en el caso de manipuladores o robots móviles estos simplemente se detendrían, un dron podría quedar seriamente dañado al precipitarse contra el suelo debido a la falta de propulsión que lo mantenga en el aire. Prever esta situación y ser capaz de incluirla entre el resto de factores en la planificación de tareas es uno de los avances que se consiguen con nuevos métodos de comportamiento basado en capas^[15].

2.1.1 Tipos de drones

La mayoría de los drones que se producen hoy en día tienen un diseño basado en vehículos aéreos tripulados reales, como aviones, helicópteros, aerostatos; o

bioinspirados, cuyo mecanismo de vuelo está directamente extraído del propio de aves o insectos. Así, se puede hacer una primera clasificación de UAVs según su diseño, en *Fixed-Wing UAS*, *Rotary-Wing UAS*, *Flapping-Wing UAS* y *Lighter-than-Air UAS*. Los FWUAS tienen dos alas fijas cada costado del cuerpo principal del vehículo, siguen el diseño y mecanismo de vuelo de los aviones, los RWUAS funcionan de la misma forma que los helicópteros con aspas rotatorias que le dan el empuje necesario para elevarse en el aire y mediante diferencias en los momentos causados por los distintos ejes de rotación consigue dirigir el vuelo.

Por lo general, los FWUAS destacan por su alcance^[18] y autonomía de vuelo, así como capacidad de carga. Sin embargo, para los casos en los cuales no se pretende realizar tareas que requieran largos vuelos ni transporte de carga, por su gran manejabilidad, se hace uso de RWUAS, ya que estos no sólo poseen un vuelo más preciso, sin requerir altas velocidades que lo mantengan en vuelo, llegando incluso a poder permanecer flotando en la misma posición, sino también la posibilidad de despegar elevándose perpendicularmente a la superficie sin necesidad de una gran espacio en el que despegar; por contra estos últimos son más ruidosos debido a las altas velocidades de los rotores. La dinámica del rotor en el caso de los FIW-UAS, se extrae del modelado del aleteo de animales voladores. Con este modelo de dron se pretende obtener el punto medio entre los dos anteriores: es una combinación de una manejabilidad aceptable y mayor autonomía y sigilo durante el vuelo que los RWUAS. Finalmente, los LtAUAS, *Lighter-Than-Air*, son drones que, como su nombre indica, tienen una menor densidad que el aire, como globos o dirigibles^[20]. Por esta razón su autonomía de vuelo es muy alta, pueden permanecer quietos en el aire durante largos periodos de tiempo sin casi necesidad de propulsión, despegar verticalmente y, por lo tanto, es una elección para aplicaciones de vigilancia en eventos masivos, en casos de desastres nucleares o químicos o incluso eliminación de minas terrestres^[23] u operaciones de rescate de víctimas de desastres naturales^[14].

2.1.2 Aplicaciones

Como se explicará en los siguientes apartados, las aplicaciones que tienen los drones pueden ser clasificados primeramente por el nivel de autonomía e inteligencia que estos posean. Así, los drones semiautónomos, los cuales suelen ser guiados por control remoto, se dedican principalmente a tareas de reconocimiento, vigilancia y adquisición de información (*RSTA*). Estas abordan, entre otras, evaluación de daños para casos de desastres naturales, recolección de muestras volcánicas, vigilancia en fronteras y cinematografía. La evolución de la inteligencia de estos drones abre un abanico de nuevas posibilidades, ya que pueden ampliar sus aplicaciones cubriendo necesidades que requieren mayores dificultades debido a entornos de trabajo más complejos, como el interior de edificios, bosques, túneles, etc. Se amplían así las posibilidades a ámbitos como las comunicaciones, el transporte o el reparto de mensajería.

De esta manera, se pueden enumerar hasta ocho tipos de tareas que, a día de hoy, están bien desarrolladas. En un primer grupo se encuentran tareas de sensorización remota, como localización fracturas en tuberías, monitorización de línea eléctrica, muestreo geológico, mapeo, meteorología, agricultura o detección de minas. Para casos de desastres naturales, existen drones usados para monitorización química de aguas y de corrientes, gestión de incendios, etc. En vigilancia se emplean para control del cumplimiento de la ley, control del tráfico o como patrullas marítimas, fronterizas o costeras. También son muy útiles en casos de búsqueda y rescate en zonas de difícil acceso. Saliendo del ámbito civil, en las industrias pueden ser usados para todo tipo de inspecciones, para transporte de cargas. En comunicaciones son una gran ayuda para ampliar el rango de señales de redes en áreas con poco alcance en comunicaciones, o como dispositivo de escucha. Gracias a la posibilidad de llevar cargas, los drones pueden incluso participar en la extinción de fuegos cargando grandes cantidades de agua o algún otro mitigador. También participar en fumigación para la agricultura. Por último, y en un ámbito más de servicios, los drones están siendo cada vez más usados para fines de cinematográficos, para el mundo académico o de entretenimiento.

Si bien se acaban de listar la mayoría de aplicaciones ya desarrolladas, a medida que avanzan las tecnologías en torno al mundo de los drones, van apareciendo nuevas aplicaciones emergentes. Por ejemplo, para las tareas de inspección en el ámbito industrial, podrían avanzar hacia casos cada vez más difíciles, empleando a veces drones en miniatura^[14] para realizar así tareas compuestas de mantenimiento, inspección y reparación de tuberías, contenedores o interactuar ahora directamente con las estructuras u objetos a reparar/inspeccionar^[19]. Aplicaciones emergentes pueden también encontrarse en torno a la agricultura, llamada ahora de precisión por los avances en sensorización que llegan a realizar mapeos y análisis de precisión sobre la calidad del cultivo, el crecimiento o para prevención de enfermedades o plagas^[10]. Finalmente, también se prevé un gran avance en las aplicaciones en imágenes y video para tareas de reconstrucción 3-D^[3].

2.2 Metodología para la Inteligencia en Robots Autónomos

Como se mencionaba anteriormente, la robótica puede ser clasificada por niveles en función de la autonomía de sus acciones. Esto es, la capacidad de poder realizar tareas con una mínima supervisión o control por parte del humano. De aquí se disgregan los robots semiautónomos, los cuales suelen ser pilotados de manera remota por un operador, de los autónomos, que tienen la suficiente inteligencia como para realizar tareas para las que fueron diseñadas adelantándose a todos los posibles imprevistos a los que puede enfrentarse en un entorno real, como obstáculos, fuerzas externas u otros, mediante sofisticadas arquitecturas de control y sensorización en tiempo real^[37]. Para conseguir esto último, el robot debe ser capaz de realizar las tareas predefinidas sin dejar nunca de recabar la máxima información posible sobre el entorno en el que se encuentra, actuando en consecuencia si fuese necesario. De todo esto se puede describir cuatro maneras de programar el control de un robot.

Programación deliberativa. Esta opción es la que permite realizar las tareas de mayor complejidad, que suelen requerir bastante de la información obtenida por los sensores y la almacenada como conocimiento. La manera de razonamiento se lleva a cabo mediante modelados, planificación y evaluación de alternativas, siguiendo una secuencia de estados y activación de acciones. Estos módulos son de gran complejidad computacional, por lo que es adecuada para tareas que no requieran un tiempo reducido o limitado de respuesta, sino más bien planificación off-line^[4]. Para que el robot sea capaz de tener de manera precisa su entorno modelado para realizar una planificación acorde a las limitaciones presentes del momento, este entorno debería ser bastante estático e invariable. Sin embargo, esto no ocurre en la práctica, ya que suelen trabajar en entornos ruidosos y dinámicos. Por esto, es difícil encontrar un robot que actúe bajo un control puramente deliberativo.

Programación reactiva. Por contrapartida a lo anterior, el control reactivo es aquel que, por necesidades de bajo o nulo coste computacional y temporal, la secuencia información-planificación-actuación se salta el paso intermedio para quedar acoplado en un comportamiento bioinspirado de estímulo-respuesta. Estos procesos no necesitan representación del conocimiento del entorno, métodos o modelos. Alcanzan así una rápida respuesta en tiempo real, apta para entornos dinámicos y cambiantes. Sin embargo, está enfocada a tareas sencillas, que no requieran cálculos complejos que consuman tiempo y puedan ser caracterizadas a priori^[16]. Cualquier otro tipo de tarea que involucre conocimiento, modelos, memoria y/o aprendizaje no puede ser resuelta por este tipo de control.

Programación Híbrida. Nace de las limitaciones de las dos anteriores. Un intento por unir las ventajas de la reactiva y la deliberativa, consiguiendo así una rápida respuesta y capacidad para memorizar y razonar a partir de conocimiento. Puesto que realmente cada una de ellas son enfoques idóneos para distintos tipos de tareas, éstas pueden entrar en conflicto, por lo que no se trata de algo trivial, sino más

bien bastante complejo. Por este motivo, la programación híbrida se organiza en un sistema de capas, cuya precisión, velocidad de respuesta y prioridad aumenta en las capas bajas a medida que decrece la inteligencia necesaria para realizar las tareas^[22]. A modo de ejemplo, un robot programado para inspección de fachadas, podría estar realizando la trayectoria dictada por su capa deliberativa y tratando las imágenes de la misma forma. Si en un momento dado aparece un obstáculo en el camino, los sensores lo detectarían y activarían la capa reactiva la cual anularía todas las tareas que se estuviesen ejecutando en ese momento para dar prioridad a sortear ese obstáculo. Sin embargo, de entre todas las maneras posibles de realizar esa tarea reactiva, elegiría una de ellas con el apoyo de la capa deliberativa que, bajo el conocimiento almacenado, le ‘aconsejaría’. Es esta necesidad de interacción constante entre ambas capas la que determina la más extendida arquitectura en este tipo de programación, la arquitectura de tres capas. Esta consiste en una primera capa de ejecución(reactiva) de bajo nivel, formada por estrechos bucles estímulo-reacción; una capa coordinativa, encargada de la comunicación y el envío de información desde la capa deliberativa a la reactiva; y finalmente una capa deliberativa, la cual genera soluciones generales a tareas complejas.

Programación basada en comportamientos. Esta alternativa distribuye el control en módulos que interaccionan entre sí para alcanzar una solución el nivel idóneo de control. Cada comportamiento (‘behavior’) recibe entradas de los sensores y/u otros comportamientos y estos, a su vez, proporcionan salidas que van a parar a los actuadores del dron o a otros comportamientos. Es una red no centralizada que representa el entorno. De manera individual, cada comportamiento puede ser un modelo, una regla básica para usar ante un estímulo, o un pequeño protocolo de comunicación entre ambos comportamientos. De esta manera, el tipo de procesamiento, tiempo de respuesta, etc. puede ir desde las propias de las más reactivas a las más complejas^[12].

El diseño de estas redes modulares es el mayor reto. Éstas son complejas debido a la necesidad de gestionar las comunicaciones entre ellas y sobre todo resolver conflictos. Hay casos de arquitecturas por comportamientos que usan selectores de comportamientos^[2], otros usan una arquitectura de tres capas en las que la reactiva está formada por comportamientos^[28].

2.2.1 Sistemas Basados en comportamientos

Estos sistemas fueron ideados para unir las ventajas de los hasta entonces conocidas y polarizadas arquitecturas. Se pretendía pues proveer al robot con la adaptabilidad suficiente para entornos dinámicos, pero sin quitarle la capacidad computacional y de memoria de los sistemas deliberativos^[25]. Con estos módulos de comportamientos permiten mantener un fuerte acoplamiento entre la sensorización y las actuaciones a través de ellas, haciendo uso de métodos tradicionales o aprendidos, y almacenando en creencias ('beliefs'), las cuales se mencionarán y describirán a fondo más adelante.

Estos comportamientos, los cuales pueden ser ejecutados de manera concurrente, actúan a modo de leyes de control y tienen una apariencia muy diversa. Pueden ser formadas por una simple ecuación que conforme una entrada y una salida, un procesamiento, un modelo completo que rijan el comportamiento de algún elemento del entorno del dron o un procedimiento que apliquen ya sea sobre software o hardware. La conexión entre ellas está determinada de tal manera que cada comportamiento que reciba una entrada(estimulo), ya sea del conjunto de sensores o de otro comportamiento, y si se cumplen las condiciones para que se active, proporcione una salida, bien a los actuadores o a otros comportamientos. Existe pues una modularización que pretende dar la mayor simplicidad posible a estos comportamientos y que sea el sistema completo el que vaya incrementando en complejidad siguiendo una filosofía 'bottom-up'. Al contrario que la metodología top-down, la metodología bottom-up comienza desde la especificación de los requisitos y capacidades de los módulos individualmente y el comportamiento final emerge de la

interacción de todos los módulos participantes y el entorno, la cual brinda la posibilidad de resolver tareas a un mayor nivel de abstracción^[20].

Una parte interesante de esta metodología es la posibilidad de tener comportamientos emergentes ('emergent behaviors'), llamados así por la capacidad de formarse a sí misma a partir de las interacciones entre otros comportamientos y en lugar de ser programado a priori. Es decir, los comportamientos últimos que pueden verse en el robot pueden ser una composición de comportamientos más simples. Esto simplifica enormemente el espacio de estados que necesita un dron para realizar infinidad de tareas. De aquí surge uno de los mayores retos en el diseño de los sistemas basados en comportamientos y es llamado comportamiento de selección o coordinación, que consiste en dotar al robot de una ley de control que sea capaz de decidir qué comportamiento o acción dentro de alguno de estos realizar.

2.2.2 Aprendizaje

Aun con todas las posibilidades que ofrecen los sistemas mencionados anteriormente para darle 'inteligencia' a un dron, siguen siendo incontable la cantidad de variables, algunas además impredecibles, que necesitaría tener el sistema para poder lidiar en un entorno real en el que la adaptabilidad se hace totalmente necesaria. Y para conseguir esta última se debe llegar a un modelo de aprendizaje adecuado.

Si bien el aprendizaje automático ha demostrado numerosas veces su utilidad y sorprendentes logros, el modo de conseguir esto para un sistema inteligente -el ajuste de ciertos coeficientes (pesos) de una red mediante la lectura de una cantidad ingente de datos- es tremendamente costosa en el tiempo, lo que lo convierte en una solución inviable para la robótica móvil, que necesita alternativas más ágiles. En estos entornos impredecibles y cambiantes, un sistema basado en comportamientos debería examinar y almacenar de algún modo la evolución de sus propios estados internos y la consecución de activación de sus módulos ante los distintos estímulos.

De esta necesidad nace las llamadas motivaciones. Estas motivaciones, junto con los anteriormente mencionados comportamientos de selección, dan lugar a un gran abanico de posibles estructuras, una de ellas ilustrada en la Figura 2.

Existen tres niveles de abstracción, en la primera de ellas de más bajo nivel se encuentran los comportamientos que unen la información obtenida del conjunto de sensorización con las acciones a llevar a cabo. El segundo nivel es el llamado *recommendation level* y está compuesto por los módulos encargados de la selección de acciones y reajustes de los parámetros internos de cada comportamiento en función de los objetivos del dron y su entorno en cada momento. El módulo llamado *External Situation* evalúa las entradas del conjunto de sensores, el módulo *needs* se basa en las motivaciones almacenadas en ese instante y el llamado *Cognition* usa el conocimiento, -obtenido a su vez del módulo motivaciones, del entorno o las intenciones computadas de todos los módulos anteriores-, para planificar o preparar un comportamiento candidato. Finalmente, el tercer nivel, *Motivation level*, monitoriza los objetivos del dron y coordina así el funcionamiento de los comportamientos. Estas motivaciones están influenciadas por el conocimiento del entorno, la información recogida en tiempo real de éste y la observación sobre la consecución de usos de los distintos comportamientos, también llamado explotación del comportamiento^[36].

Estos sistemas ‘motivacionales’ han conseguido equilibrado la planificación/computación y la reactividad para poder gestionar adecuadamente las tareas requeridas usando variables internas que se activan o desactivan mediante diversos factores. Existen cuatro enfoques principales para el aprendizaje que han conseguido validar con éxito esta metodología.

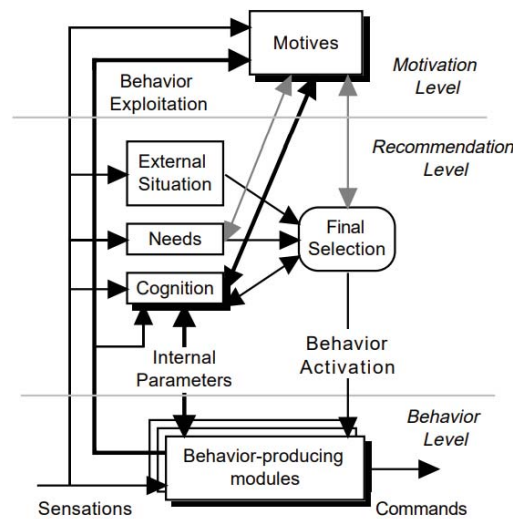


Figura 2. Representación de un posible sistema de 3 capas

Aprendizaje por refuerzo. Una de las técnicas más conocidas de aprendizaje en el mundo de la robótica han conseguido integrarse sorprendentemente bien con los sistemas basados en comportamientos. El aprendizaje por refuerzo se basa en un sistema por prueba y error de recompensas que el robot obtiene según la acción tomada en cada instante. Así, al conseguir su objetivo, éste es capaz de cuantificar estas recompensas obtenidas durante el proceso e intentar optimizar la consecución. El principal límite de este tipo de aprendizaje es el incremento exponencial de la complejidad y tiempo de computación con la dimensionalidad. Sin embargo, al convertir el inmenso conjunto de variables que compone un entorno en comportamientos, estas se reducen considerablemente. Esta metodología ha podido ser validada con éxito numerosas veces, entre ellas en el aprendizaje de un hexápodo a caminar^[9] y un robot OBELIX empujando cajas^[26].

Aprendizaje mediante redes. Esta opción modula en forma de red los comportamientos y sus conexiones. Aquí se divide el comportamiento en *abstractos* y *primitivos*. Las primeras separan las condiciones de activación de sus acciones de salidas, gestionadas por las últimas. Las redes quedan formadas por estos

comportamientos abstractos, representados como nodos, y las interdependencias condicionantes entre estas, como arcos. Mediante estas redes se ha conseguido introducir un generador automático de descripciones de tarea que trabajaba mientras observa el entorno. Así, pudo ser validado en un robot móvil que seguía a una persona e iba adquiriendo la representación de las tareas que éste realizaba. El robot observaba la secuencia de activaciones que realizaba y de ahí obtenía una representación mediante nuevos comportamientos abstractos^[32].

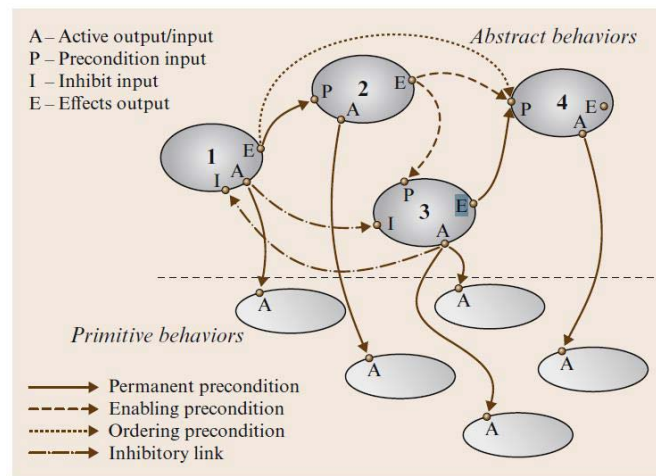


Figura 3. Representación en grafo de un sistema de aprendizaje basado en redes

Similar al anterior, el *aprendizaje por demostración*, ha conseguido integrarse con estos sistemas de comportamientos mediante instructores que a priori le indican la secuencia de comportamientos a seguir para determinadas tareas. Mediante diversas técnicas, como por ejemplo estimación de estados, el robot generaba las interdependencias entre estas activaciones.

Finalmente, el *aprendizaje por historial de uso de comportamientos*, usa información de históricos para obtener secuencias temporales del uso de estos comportamientos para tomar decisiones con inteligencia. Esta metodología se puede dividir en: un mecanismo de generación de las secuencias en árbol, evaluación del

rendimiento de las secuencias, un criterio de recomendación de alternativas y un mecanismo de eliminación de caminos. Para cada tarea existe una representación que comienza vacía y se van añadiendo nodos con la información del tipo de comportamiento usado y el número de veces que se ha pasado por ese camino. Cada vez que se reúsa se actualiza el rendimiento mediante las métricas del siguiente paso. Esta metodología demuestra que el robot puede aprender estrategias para saber qué debe obtener del entorno y así aprender a optimizar la tarea, proceso idóneo para aprender en entornos no-estacionarios [21].

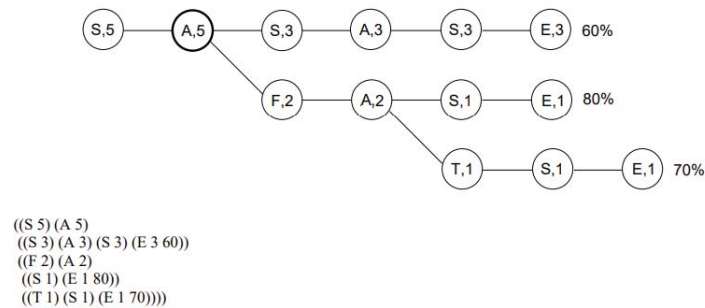


Figura 4. Posible árbol generado por aprendizaje

2.3 Adquisición de datos y sensorización

Un elemento clave para que todo lo descrito hasta el momento sea posible es la capacidad de obtener información del entorno en tiempo real. Si bien desde sus comienzos, la robótica se ha ayudado de múltiples tipos de sensores, desde los más tradicionales como sensores de contacto, de presión, etc., hasta las más novedosas técnicas de sensores de distancia por láser, pasando por los avances con geolocalización GPS-IMU, esta sección se centrará especialmente en aquellos que, por su viabilidad en espacios cerrados, se convierte en crucial para los RWUAVs. Estos son los llamados en inglés, *range sensing*.

2.3.1 Sensores de proximidad, contacto y fuerza

La sensorización táctil o de contacto lleva bien establecida casi tanto tiempo como la robótica misma. Los primeros sensores de contacto, por respuesta mecánica con un botón, detectaban la presencia del objeto cuando éste presionaba el botón al entrar en contacto. También para un nivel propioceptivo, es decir, aquellos sensores encargados de conocer el estado cinemático y dinámico interno del robot, de cada una de sus articulaciones, existe tecnología muy bien establecida, como potenciómetros, encoders, resolvers, etc.

Este tipo de sensorización es de gran importancia en el caso de robots que llevan a cabo tareas delicadas que requieren alta precisión. La gran información que en la naturaleza provee el tacto puede limitar incluso el uso a otros grandes competidores en la sensorización como puede ser la visión. Existen numerosos ejemplos en la práctica en los que no podríamos clasificar un objeto por visión y sí por el tacto, puesto que las texturas pueden ser en muchas ocasiones engañosas a la vista. Sin embargo, los sensores táctiles se encuentran con retos mucho mayores que ésta última, y de ahí su mayores logros y desarrollo. El ser humano posee una cantidad de mecano receptores en la piel del orden de cientos o miles por centímetro cuadrado, lo que hace prácticamente imposible encontrar una solución a un posible cableado. También trabaja con enorme resistencia a la fricción, el contacto o los golpes. A pesar de esto, la robótica siempre avanza en la misma dirección que la naturaleza y se ha conseguido importantes avances como por ejemplo con sensores antena imitando el comportamiento y uso de los bigotes de los mamíferos o las antenas de algunos insectos^[8].

Los sensores de proximidad sí que han sido mucho más estudiados y usados, principalmente para sortear obstáculos y evitar colisiones. Existen numerosos casos de sensores capacitivos de proximidad que hacen uso de osciladores electrónicos para analizar la variación de frecuencia en el campo dieléctrico, que sería el entorno a detectar, y de ahí estimar la distancia al objeto. Se ha desarrollado complejos algoritmos de evitación de humanos en robots móviles mediante esta tecnología^[30].

También han proporcionado grandes avances en sensores las tecnologías de infrarrojos y fibra óptica[7,6,33]. Dentro de este grupo de sensores se encuentra también los sensores térmicos que, en la mayoría de los casos mediante materiales piezoeléctricos, son capaces de detectar contacto o cercanía a ciertos objetos. Las investigaciones más avanzadas incorporan termistores en pieles artificiales[5]. Otros sensores se enfocan en la detección de la composición del material mediante sensorización de fase liquido-vapor[24], o mediante piezas electromagnéticas aprovechando el efecto de Hall[31].

Lo más inmediato para tener información sobre la carga o el esfuerzo ejercido por un actuador es mediante la medición de la corriente del servomotor que lo dirige. Sin embargo, debido a las pérdidas que ocasionan el sistema de alimentación de los robots, es más preciso medir la torsión aplicada mediante la lectura óptica o electromagnética de la deformación sufrida por algún material. Se ha propuesto sensores de seis grados de libertad compuestos por condensadores eléctricos y elastómeros haciendo las veces de dieléctrico[1].

2.3.2 Sensores inerciales

La odometría consiste en el uso de información de posición de un robot para conocer su estado y predecir o estimar el movimiento del mismo. Entre estos pueden estar algunos sensores internos, los cuales, como la mayoría, aprovechan fenómenos físicos para obtener información, como los giróscopos y acelerómetros, mientras que otros obtienen los datos directamente de la posición relativa con respecto a satélites espaciales, como es el caso del conocido GPS.

Los primeros giróscopos ya habían sido patentados en el siglo 19. Estos se apoyan en el principio de conservación del momento angular. Están formados por péndulos que pueden girar en cualquier eje por lo que seguirá fielmente los cambios de orientación. Como limitación, estos sensores no apuntan directamente a la dirección de rotación, sino que fluctúan sobre ellos. Otros en cambio usan el llamado

efecto Sagnac, que calcula las diferencias en el recorrido circular de luces pulsadas en direcciones opuestas, en el mismo instante^[13]. Existen varias técnicas para medir estas diferencias, como estudiando la variación Doppler o la frecuencia de pulsación entre ambas señales. La mayoría de sensores usados en microelectrónica aprovechan las vibraciones y resonancias surgidas de la rotación debido a las fuerzas de Coriolis.

Pasando a mencionar los acelerómetros, si bien estos se usan para medir las fuerzas externas que se aplican sobre él, no se trata de un sensor de fuerza como cabe esperar. Una de las fuerzas externas a las que todo cuerpo es constantemente sometido es la gravedad, por lo que se puede conocer la posición de un objeto en altura e incluso orientación, gracias a este tipo de sensores. Según el fenómeno aprovechado para detectar se puede diferenciar entre acelerómetros mecánicos y piezoeléctricos.

A un sistema formado por giroscopios y acelerómetros para obtener valores de la posición, velocidad y aceleración en conjunto se le llama unidad de medida inercial (IMU). Mediante tres giróscopos ortogonales y otros tres acelerómetros se obtienen las medidas de las aceleraciones angulares y radiales, respectivamente. Mediante la integración de estos valores relativos se obtienen las velocidades y posiciones/orientación.

Pero sin duda, el sensor de localización más usado en el mundo es el GNSS (*Global Navigation Satellite System*), o más conocido por el nombre propio del sistema americano GPS (*Global Positioning System*). El sistema GPS está basado en el cálculo de desfase entre las recepciones de señal por diferentes satélites. Así, por triangulación, el sistema es capaz de proveer una posición en coordenadas absolutas en cualquier parte del mundo con un error máximo de alrededor de veinte metros. Otros sistemas GNSS en el mundo son GLONASS (*Globalnaya Navigatsionnaya Sputnikovaya Sistema*), del gobierno ruso y *Galileo* el cual no es militar. A pesar de su amplio uso en dispositivos electrónicos, depender de una señal microondas satelital se vuelve un problema al enfocarse en robots móviles, los cuales necesitan, en muchos casos, trabajar en recintos cerrados donde este tipo de señal no alcanza.

2.3.3 Sensores de rango

Los sensores de rango, de radio o distancia, son dispositivos capaces de reconstruir el “mundo” inmediato que les rodea en 3D. Este es el más importante de los sensores en robótica ya que mediante esta reconstrucción, el robot es capaz de ir hacia objetivos, evitar obstáculos, seguir rutas, etc. El método más común para esto es mediante luz, que recoge la posición y distancia de ciertos puntos y, mediante conversiones de coordenadas y parámetros intrínsecos del sensor, se *reconstruye* la imagen proyectada.

Estos sensores pueden clasificarse según la metodología empleada para el cálculo de la distancia. Una de ellas la obtiene a partir de dos sensores separados una distancia determinada; si el objetivo forma un ángulo recto con uno de los sensores, el ángulo formado por el otro estará directamente relacionado con la profundidad, la distancia entre el primer sensor y el objetivo. Por otro lado, los llamados TOF (*Time of Flight*) cronometran el tiempo que tarda el pulso en volver tras la refracción.

Triangulación

Estos sensores encuentran su primera limitación en la misma magnitud que pretenden calcular, la distancia. Por trigonometría, al recibir la imagen con los dos haces iluminando una superficie, se puede calcular la distancia a esta mediante la distancia entre los dos puntos del láser o luz, conociendo el ángulo del sensor secundario y el focal de la cámara. Pero cuanto más lejano sea el punto a calcular, menos resolución tendrá. De hecho, esta precisión cae con el cuadrado de la distancia a calcular. Esto puede manipularse con la distancia entre sensores o el FOV.

Mediante esta misma técnica es posible realizar reconstrucciones completas tridimensionales de nuestro alrededor^[29], materia la cual no es objeto en esta memoria.

TOF (Time Of Flight)

Como se dijo anteriormente, estos tipos de sensores siguen la misma ley del radar. Sin embargo, medir directamente el tiempo de respuesta de una emisión que viaja a 300000km/s requiere una elevadísima precisión, por lo que existen alternativas, cálculos indirectos más viables. Existen algunos como LIDAR o LADAR que lo miden así, aunque emplean varias medidas para hacer uso del promedio ya que, no sólo la precisión necesaria es un problema, sino también otros factores como la distancia que reduce la intensidad de la señal y puede obtener una señal de respuesta muy leve. El hecho de enviar varios pulsos de luz, suaviza algunos errores a la vez que incorpora nuevos factores a tener en cuenta y que limitan su uso. Por ejemplo, el PRI (*Period Repetition Interval*) debe analizarse junto al PW (*Pulse Width*) de tal manera que no exista ambigüedad entre respuestas consecutivas.

Existen modelos en los que se envían múltiples pulsos de luz con cierta frecuencia de tal manera que se consiguen grandes resoluciones. Más que incluir más y más emisores, estos sensores contienen pequeños conjuntos de motores y espejos que multiplican la señal y realizan barridos, pudiendo alcanzar los veinticinco mil puntos por segundo. Otra alternativa para multiplicar las lecturas es como las realiza el flashLIDAR, el cual, en lugar de un haz de luz, posee un LED que cubre un área determinada y, mediante un panel formado por arrays de receptores similar a las cámaras, detecta la respuesta en cada localización, extrapolando esta respuesta a un pixel de la imagen.

Los sensores que hacen uso de esta tecnología de manera indirecta, como bien dice la palabra, aprovechan la variación de algún otro parámetro más fácilmente medible provocado de manera indirecta por el desfase de tiempo. Comúnmente hacen uso de señales de moduladas en amplitud o frecuencia. Con las primeras, la distancia puede ser calculada a partir del cambio de fase de la señal recibida, en el segundo caso, la señal varía su frecuencia en forma de diente de sierra con una frecuencia máxima, y la señal recibida lleva una frecuencia igual a la diferencia de la emitida y la de respuesta, la cual puede ser medida fácilmente y de ahí la distancia.

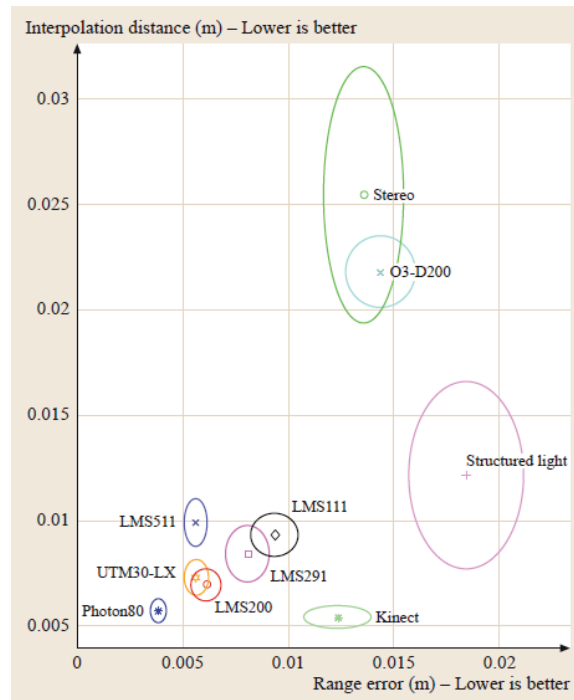


Figura 5. Comparativa gráfica de sensores de rango

DESCRIPCIÓN DEL PROBLEMA

Las secciones anteriores han realizado una descripción básica sobre la robótica aérea, que puede permitir entender la complejidad que supone su diseño y saber a rasgos generales en qué se basan los sistemas que dotan a éstos de inteligencia como para realizar tareas de un modo autónomo, así como las principales líneas de investigación existentes en este campo. A continuación, como siguiente punto importante del trabajo, se detalla el problema en cuestión, que es el análisis de utilidad y viabilidad de una herramienta software como VREP para simular misiones creadas en Aerostack.

Primero, se enfatiza la necesidad creciente de que existan software de simulación, cómo esto ayuda a los ingenieros en su diseño, a la viabilidad de los proyectos y, de manera particular, cómo se está usando cada vez más en el campo del aprendizaje automático para profundizar, después, sobre qué se busca conseguir u obtener de este estudio sobre VREP.

3.1 Importancia verificación

Como se adelantaba en las primeras secciones de la memoria, la verificación o *testing* es una fase en la ingeniería robótica que permite al ingeniero comprobar que el trabajo que ha desarrollado en cada una de sus fases cumple con los requisitos planteados. Es decir, para el caso presente, supóngase que un investigador del grupo CVAR elabora una nueva misión formada por un nuevo comportamiento desarrollado que, por ejemplo, hace que el dron compruebe su estado de carga de manera periódica

y realice una serie de cálculos y predicciones en función a las tareas realizadas y las trayectorias recorridas hasta el momento, para elegir el momento óptimo en el que debería ir a la estación de carga y recargarse. En esta situación, el siguiente paso sería probar esto integrándole el programa al dron y ver cómo actúa. Podrían darse algunos escenarios: que todo saliese tal y como se esperaba y realizase la misión acorde a lo preestablecido encontrando un momento para cargarse y seguir con la tarea, o que no saliese bien y, por ejemplo, no encontrase un momento adecuado para cargarse y siguiese realizando la tarea hasta agotar la batería y precipitarse contra el suelo. Estos drones pueden llegar a tener un elevado coste (por ejemplo, varios miles de euros). Teniendo una idea de las complejas arquitecturas de control que van incorporadas en estos drones, no parece nada improbable que pueda ocurrir algo inesperado que haga que el robot no se comporte como se espera. No sólo un descuido a la hora de programar lo nuevo, en el algoritmo en sí, las comunicaciones entre comportamientos, etc., sino también cualquier factor externo al dron propio del entorno, podría ocasionar un error en la tarea y, muy probablemente, que éste tuviese consecuencias económicamente negativas. Sin embargo, las herramientas de simulación permitirían diseñar el robot siguiendo el modelo real tan fielmente como se desee, así como el entorno en el que se probaría. La simulación del entorno y el robot puede tener en cuenta infinidad de variables y cualidades que dotan el conjunto en una simulación casi perfecta en la que, factores como las masas, centros de gravedad, momentos inerciales, rozamientos entre muchos otros, entrarían en juego. En caso de que la prueba fallase, el ingeniero puede modificar lo que vea conveniente sin salir del ordenador ni levantarse de la silla y, por supuesto, eliminando cualquier posible consecuencia desastrosa que desencadenase algún error en la ejecución de la misión.

Éste ha sido tradicionalmente la ventaja principal de los simuladores en robótica. Sin embargo, desde hace unos años la creciente popularidad que experimenta el campo de la inteligencia artificial, en especial el aprendizaje automático, ha encontrado también una gran ventaja en las simulaciones.

Aunque no entraremos en profundidad a detallar en qué consiste el aprendizaje automático, de anteriores apartados se conoce un poco sobre los tipos de aprendizaje más usados en robótica. Así, se entenderá el tiempo que requiere un proceso completo de aprendizaje. Comparado con el aprendizaje automático supervisado de las redes neuronales, por ejemplo, el aprendizaje por refuerzo puede ser muchísimo más costoso en tiempo. En el proceso llamado entrenamiento del primer método, usado por ejemplo para clasificación de imágenes, el sistema recorre una base de datos en las que el sistema recoge todas las entradas a partir de las cuales debe generar una salida que será comparado con la salida deseada para ajustar, en función de error, varios parámetros que, después de miles de ejemplos vistos concluye en una red capaz de clasificar correctamente. Estos sistemas, en función de la complejidad del problema, suelen requerir del orden de cientos de miles de ejemplos vistos, en algunas ocasiones realizando varios ciclos (llamados épocas). Sin embargo, por muy inmenso que sea el conjunto de datos a recorrer, las grandes capacidades computacionales que hay disponibles hoy en día hacen que estos procesos puedan llevarse a cabo en cuestión de horas o días. En el caso del aprendizaje por refuerzo, el agente debe realizar una tarea completa, con todos sus pasos intermedios, hasta llegar a la resolución de la misma, momento en el cual recibe una recompensa en función de la efectividad de la solución. Si nos vamos al caso que atañe y se pretendiese que un dron aprendiera, por ejemplo, la secuencia correcta de comportamientos a activar para realizar cierta misión, el sistema debería realizar esta tarea, como mínimo, tantas veces como posibles combinaciones de comportamientos existiese. Con que sólo hubiese disponible ocho comportamientos y limitando la secuencia a diez comportamientos, se debería realizar la tarea ocho elevado a diez = $1.073.741.824$ veces. Aunque la tarea requiriese sólo 20 segundos, se tardaría en probar todas las alternativas posibles en unos 680 años. Pero, ¿qué pasaría si fuese posible reducir esta tarea a mili segundos? Esta es la razón por la cual la simulación virtual se vuelve tan valiosa para el aprendizaje. Otra ventaja de la simulación viene de recientes pruebas y métodos de aceleración del entrenamiento realizadas con aprendizaje en paralelo por varios robots^[35].

Ante este interés, VREP se convierte en un buen candidato para ser usado como simulador de los sistemas inteligentes creados en Aerostack, pero es necesario antes analizar su viabilidad técnica para conocer, de manera cuantitativa y cualitativa, su desempeño.

METODOLOGÍA DE EVALUACIÓN

Este apartado introduce al siguiente detallando los factores y variables que se han tenido en cuenta para la evaluación de VREP como software de simulación para Aerostack. Sin entrar en el proceso seguido para la evaluación o la construcción de escenarios, se hace hincapié en la manera en que evaluaremos posteriormente el mismo.

Para conseguir esto, se comienza describiendo las herramientas involucradas, VREP y Aerostack, para que el lector vaya con un conocimiento básico a los posteriores apartados donde se hablará más sobre estos. Una vez hecho esto, se enumeran las variables y evaluadores elegidos: en qué consisten, la razón que lleva a elegirlos y la visión que proporcionan sus resultados. Finalmente se describe, paso a paso, el orden que se seguirá en esta evaluación.

4.1 Software empleado

La herramienta principal, Aerostack, es un software de código libre en continua construcción, por lo que puede ser muy vulnerable en cuanto a compatibilidades con otros componentes software. A continuación, se desarrollan sus principales funcionalidades.

4.1.1 Aerostack

Esta herramienta es el resultado de varios años de investigación y desarrollo en el campo de la robótica por parte del grupo CVAR de la Universidad Politécnica de

Madrid. Es un entorno software, ejecutado sobre ROS (*Robot Operative System*) como middleware, que brinda las herramientas necesarias a los desarrolladores para crear las arquitecturas de control de UAS. Es un software de código libre (GPL) que permite, no solo construir sistemas aéreos autónomos, sino desarrollar nuevas soluciones y evaluarlas.

Esta herramienta proporciona una librería de componentes software escritos en C++ y Python, así como la posibilidad de añadir nuevos componentes. Entre estos componentes, hay soluciones como control de movimiento, estrategias de planes de vuelo, extracción de características mediante visión artificial, SLAM, etc. Los desarrollado o creado con Aerostack, puede ser probado en vuelos reales o en Gazebo o VREP, simuladores compatibles con la herramienta.

Las misiones pueden ser ejecutadas desde un ordenador a bordo, mediante red local LAN o WLAN para sistemas multirobot. Así, a partir de lo leído anteriormente, se pueden probar diferentes misiones basadas en complejas arquitecturas de control, añadiendo comportamientos a las distintas capas y testeando éstos en alguno de los simuladores o en un vuelo real.

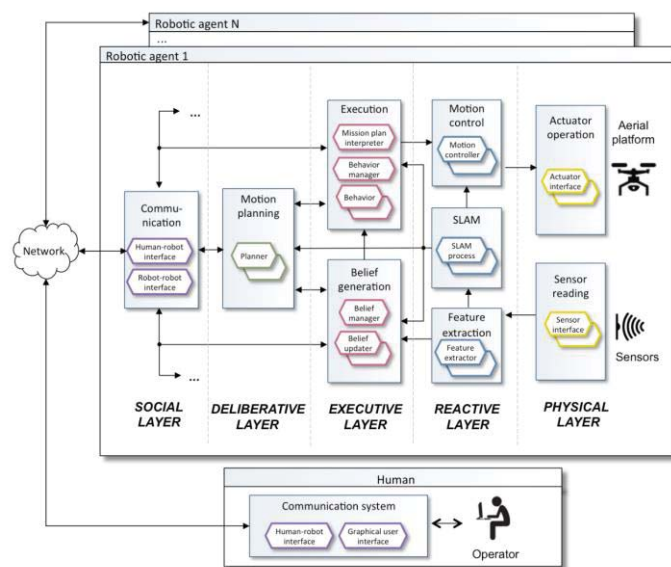


Figura 6. Esquematización de la arquitectura de control de Aerostack

La arquitectura de la herramienta Aerostack para su uso, sigue el modelo de sistema híbrido deliberativo/reactivo en varias capas: ejecutiva, reactiva, deliberativa, social y física.

Las tres primeras capas siguen la metodología de la así llamada *three-layer architecture*, la cual se compone de una reactiva (*controller*), encargada de responder a modo acción-efecto los estímulos recibidos de los distintos sensores para realizar tareas sencillas que permitan una ejecución rápida que no obstruya el lazo de control cerrado; la ejecutiva (*sequencer*), a la que se le brinda un poco más de capacidad de memoria, de tal manera que no solo responde a un estímulo sin tener nada más en cuenta sino que es capaz de almacenar secuencias anteriores, actuando de selector de comportamientos y nexo de unión entre las otras dos capas; y la deliberativa, que no tiene límites de memoria o tiempo de computación y sirve para tareas de predicción, planificación y procesamiento. Así, por ejemplo una secuencia de trabajo de las tres capas ante una tarea común sería la siguiente: en un laberinto, la capa reactiva detecta un muro y frena o cambia de dirección, envía el punto a la capa ejecutiva que ordena a la primera a seguir el punto que tuviese almacenado; cuando llega a un callejón sin salida, la capa ejecutiva lo manda recorrer todo el camino anterior de vuelta y envía a la capa deliberativa el recorrido para que lo almacene en mapa, lo procese y planifique una nueva alternativa de ruta.

Con respecto a la capa social, esta se integra para encargarse de las comunicaciones en sistemas multiagente.

Estas tres capas trabajan por separado, lo que se traduce en gran ventaja debido a los largos tiempos de procesamiento que no pueden permitirse las capas más livianas.

Cada componente de la herramienta(comportamiento) es llamado proceso(*process*), que a su vez se agrupan y se conectan entre ellos para formar sistemas. Esta modularidad brinda la posibilidad de seguir desarrollando nuevos procesos o sistemas sin la necesidad de cambiar nada por debajo, incluso hacerlo puramente en C++, por ejemplo, sin que ninguna librería de ROS, se vea involucrada.

Cada proceso es ejecutado desde un fichero *.launch*, que es interpretado como un nodo de ROS y puede correr concurrentemente usando el MOM (*Message-oriented middleware*) en modo publicación/suscripción.

En resumen, estos procesos se diseñan y agrupan para crear sistemas. Cada sistema o grupo de sistemas pueden ser empaquetados para usar en una determinada misión, la cual requiere un *launcher*. Un launcher es un archivo *.sh* que contendrá toda la información necesaria para ejecutar una misión. Además del launcher, éste estará asociado a un directorio que contendrá toda la configuración, la cual incluye la plataforma de la misión, los agentes, el plan de la misión, etc. El siguiente paso es crear un entorno de simulación en el cual se quiere llevar a cabo la misión.

4.1.2 V-REP

Como se ha mencionado con anterioridad, un simulador de robótica es una aplicación que permite programar, emular y validar un robot sin necesidad de su presencia física, ahorrando así costes y tiempo. Básicamente, es un software con un entorno gráfico para crear objetos, materiales y, en general, un mundo virtual donde validar los comportamientos programados para transferirlos al robot real. Entre los más usados por la comunidad robótica se encuentra V-REP de la empresa Coppelia Robotics. Este software multiplataforma (perfectamente ejecutable en Linux, MacOS o Windows) dispone de un IDE propio para construir escenarios virtuales y controlar cada robot mediante scripts internos escritos en LUA, nodos ROS o programas externos escritos en lenguajes como C++, Matlab, Octave, Java y Python (entre otros). Esta versatilidad que ofrece permite que pueda ser usado como simulador para verificar los procesos/sistemas/misiones creados en Aerostack.

La interfaz gráfica ofrece una serie de modelos de robots u otros elementos ya diseñados (normalmente importados desde algún software CAD), así como todas las funcionalidades y facilidades necesarias para crear tu propio modelo. Así, puedes importar, si existe, el modelo del robot que quieres validar o crearlo gracias al modo de edición de formas que ofrece la herramienta, mediante esto y la adecuada

ordenación de sus componentes en el marco de *jerarquía de escenario* se pueden diseñar robots tan *complejos como se desee*.

La Figura 7 muestra la interfaz gráfica de usuario principal. En esta se encuentra centrada el entorno virtual de simulación o ventana de aplicación, vacía inicialmente. A la izquierda un explorador de modelos a partir del cual podemos arrastrar e incluir los modelos disponibles en el entorno virtual de simulación. Se encuentran agrupados por tipo y se pueden encontrar desde conocidos robots comerciales hasta mesas de oficina o terrenos de exterior. Junto a esta se encuentra la jerarquía del escenario, que muestra todo el contenido del escenario (conjunto de todos los elementos) en forma de árbol jerarquizado. Esto es así ya que los elementos añadidos pueden configurarse como *hijos* de otros, con las consecuentes propiedades que se explicarán más adelante. Cada elemento incluido en escenario, se verá en la jerarquía del escenario incluyendo el tipo de objeto mediante un icono, el nombre del objeto, un icono con el que abrir el script asociado y otro para la lista de parámetros. Además de esto, las barras superiores posibilitan al usuario a navegar por el entorno gráfico, modificar la posición y orientación de cada objeto, o modificar ciertos parámetros de simulación.

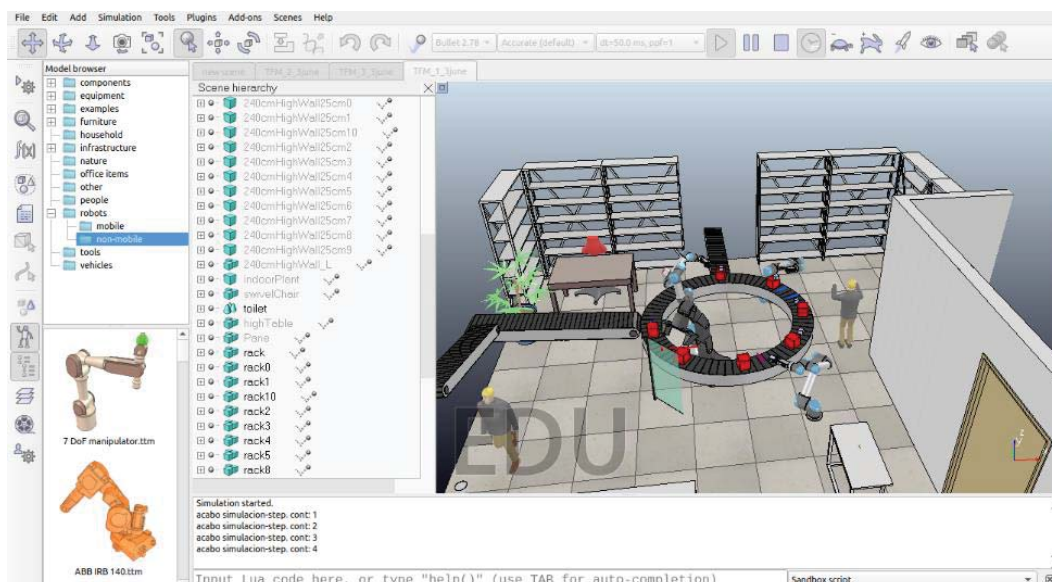


Figura 7. GUI de VREP. Ventana principal

4.2 Aspectos a evaluar

Aerostack es un software que se ejecuta en Ubuntu, de Linux. Si bien se encuentra en continua mejora de tal forma que se actualiza para estar siempre disponible para la última versión LTS de Ubuntu, en este caso con la versión 18.04, no queda totalmente garantizada el correcto desempeño en todos y cada una de sus posibilidades. Usa ROS (Robot Operating System) como middleware que se encarga de las comunicaciones con otras aplicaciones, como puede ser VREP, aunque también Gazebo.

Por esta razón, este apartado puede subdividirse en dos en función del tipo de análisis. El primero estará dedicado a analizar los aspectos más técnicos que permitan sacar resultados cuantificables. Uno de ellos es la facilidad de integración con Aerostack, lo amigable que sea el proceso de instalación, así como la integración y configuración de las comunicaciones con Aerostack, los cuales serán determinantes en la evaluación de VREP. También su funcionalidad en cuanto a las prestaciones y posibilidades que ofrece la herramienta para crear escenarios de simulación, que será un análisis basado en la experiencia tras crear tres escenarios. Y finalmente, se estudiará el factor eficiencia a partir de una serie de pruebas basadas en el tiempo de respuesta de la herramienta, de sus exigencias en cuanto a los recursos computacionales que consume (memoria, CPU, tarjeta gráfica, etc.). Como se venía comentando, una ventaja valiosa de este tipo de herramientas es la posibilidad de poder simular muy por encima del tiempo real para tareas de aprendizaje automático. También esto se valorará, comprobando cuántas veces más rápido se puede acelerar, cómo afecta a la memoria del equipo y si alcanza un mínimo para que pueda considerarse una herramienta atractiva y útil.

Además, se argumentará a nivel cualitativo ciertos factores a tener en cuenta, como puede ser la facilidad de uso a la hora de construir los escenarios donde se simule. Dentro de este factor, pueden quedar desglosados otros subaspectos como la sencillez de la interfaz gráfica, la documentación disponible acerca de la herramienta, la soltura con la que se pueda navegar a través de un escenario y mover objetos, agrandarlos, y el tiempo incurrido en todas estas tareas. Otros aspectos sobre los que

se discutirá estarán relacionados con las facilidades que ofrece ésta, es decir, lo cómodo que resulta la herramienta a la hora de poder acceder a ella (precios, licencias, etc.), y el mantenimiento y la portabilidad que ofrece.

Enlazado con esto y orientados por la norma ISO/IEC 25000 SQuaRE (*System and Software Quality Requirements and Evaluation*) de calidad software^[15], se pueden enumerar los siguientes indicadores de calidad:

- *Funcionalidad*. Entendiéndose ésta como el conjunto de atributos que se relacionan con la existencia de las funciones y sus propiedades específicas. Puede venir indicado por la capacidad para cubrir todas las necesidades que puedan existir bajo cualquier escenario posible desde Aerostack. Es decir, cualquier requisito que pueda surgir de un nuevo desarrollo en esta herramienta.
- *Confiabilidad*. Conjunto de atributos relacionados con la capacidad del software a mantener su nivel de rendimiento bajo ciertas condiciones y periodos de tiempo determinados. Se analizará este indicador junto al de eficiencia por sus similitudes. En este apartado se incluirá el análisis de integrabilidad con el resto de componentes.
- *Eficiencia*. Conjunto de atributos que definen el ratio entre el nivel de rendimiento y la cantidad de recursos usados, bajo ciertas condiciones. Bajo este indicador, se realizarán pruebas mediante simulaciones repetidas en distintas situaciones: número de agentes en la escena y velocidad de simulación, mediante el número de ejecuciones del main por paso de renderizado. La herramienta VREP tiene una opción que permite aumentar y reducir la velocidad de simulación. No indica en cuánto está aumentando y sólo permite aumentar hasta tres veces.

- *Usabilidad.* Conjunto de atributos que tienen que ver con el esfuerzo requerido para su uso general o usos concretos. Este indicador viene medido por la facilidad de uso, la amigabilidad para con el usuario de la interfaz gráfica.
- *Mantenibilidad.* Bastante relacionado con la usabilidad, pero refiriéndose esta vez al esfuerzo que supone crear modificaciones, séanse principalmente éstas, sus propias actualizaciones. De manera más cualitativa, se exponen las impresiones recogidas desde la propia experiencia con la herramienta.
- *Portabilidad.* La capacidad del software para pasar de un entorno a otro, así como su accesibilidad al público. De la misma forma que con la mantenibilidad, se comenta este atributo de portabilidad de VREP.

CONSTRUCCIÓN DE ESCENARIOS DE PRUEBA

Los tres escenarios contruidos en VREP son el tema principal de la primera parte de este apartado. Se describirán en profundidad, enumerando la mayoría de objetos de relevancia de cada escenario, explicando su utilidad y dando una visión general del aprovechamiento que se espera obtener de cada uno de ellos.

Posteriormente, y descrito en el anterior apartado el criterio que sigue el análisis de grado de adecuación de la herramienta VREP para validar misiones realizadas en Aerostack, se describe detalladamente en este apartado el procedimiento seguido, así como sus resultados.

5.1 Descripción de escenarios

La herramienta Aerostack posee arquitecturas de control para drones que permiten llevar a cabo tareas tales como perseguirse entre ellos, seguir instrucciones a partir de lecturas de marcas visuales ArUco, realizar inspecciones de procesos industriales o inventario, etc. Todo esto siguiendo el paradigma descrito en la sección pertinente con respecto a las distintas capas y comportamientos.

Así, se han diseñado tres entornos virtuales con los que se podrán probar diversas misiones, ya sea con el uso de un solo dron o multiagente. Este desarrollo ha sido llevado a cabo siguiendo las necesidades y requisitos de las actuales misiones, ya desarrolladas o en desarrollo, en las que trabaja el grupo de investigación Vision4UAV. El trabajo se ha centrado en la futura validación de las siguientes misiones: Persecución de drones, Inventario de almacén e inspección de calidad de procesos.

5.1.1 Escenario 1. Persecución

Este escenario ha sido diseñado para probar misiones de tipo persecución. En él, un robot móvil ‘objetivo’, con una trayectoria programada, es perseguida por el dron bajo prueba. Éste debe llevar programado un sistema inteligente que le permita llevar un control de la trayectoria y velocidad según las entradas de los sensores, así como otros comportamientos de tipo reactivo que eviten que se caiga o se accidente contra algún elemento. Se ha elegido un escenario outdoor, como primera prueba antes de una definitiva que sería llevada a cabo en un recinto cerrado. Puesto las funciones *follow-me* actuales en el mercado están preparadas para funcionar en el exterior gracias a lecturas recogidas de GPS, esta misión sería a partir de sensores de rango ya que, como es bien sabido, la señal GPS no puede introducirse en la mayoría de recintos cerrados. Existen numerosos métodos alternativos para realizar tracking, ya sea por tratamiento de imágenes, por señal RF, por señal acústica, etc. [\[17\]](#).

Por otro lado, existe el tema del continuo multipath: tras el reconocimiento del objeto a seguir y el cálculo de la distancia, el dron debe planificar una ruta hacia él, tarea con un número de soluciones alternativas que crece a medida que el número de objetos y obstáculos entre medio aumenta también. Si a esto se la suma el hecho de que ambos agentes estarán continuamente en movimiento por lo que deberá realizar esto de manera iterativa y continua, convierte la tarea en un proceso con un coste computacional bastante grande. El dron deberá estar preparado para que su capa reactiva se active ante la presencia cercana de obstáculos, reaccione y envíe su posición a la capa ejecutiva que, a su vez, solicite a la capa deliberativa un nuevo path planning teniendo en cuenta la última información.

El escenario está compuesto por un objeto de tipo suelo llamado que emula los desniveles propios de terraplenes, demás objetos que típicamente podríamos encontrar en el exterior, como árboles y matorrales, rocas; y finalmente personas caminando, distribuidos tal y como muestra la Figura 8.

Bajo estas condiciones, el dron debería ser capaz de trabajar concurrentemente de manera que realice actualizaciones de la ruta planificada continuamente a partir de la distancia, localización y orientación del objetivo. A esto se le suma el cálculo de la velocidad y aceleración que deberá llevar para mantener las distancias. Durante esta misión, deberá también estar capacitado para sortear los obstáculos que, debido a la distancia con el objetivo y a su trayectoria irregular, se encuentren de manera inesperada por su camino.

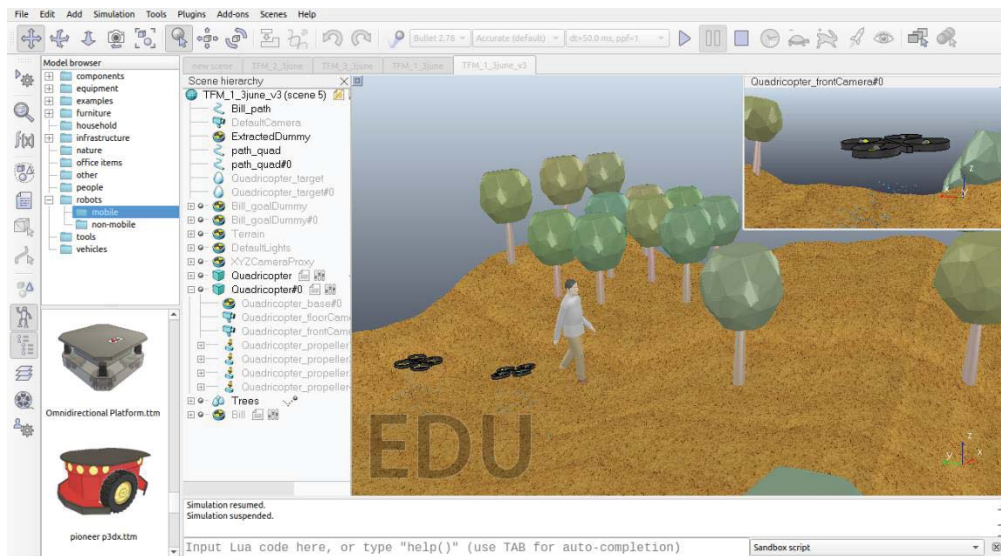


Figura 8. Escenario para persecución de drones

5.1.2 Escenario 2. Inspección

Este escenario es el más elaborado y complejo, el cual coincide con la Figura 7. Lo fue tanto para su creación, diseño y programación, como lo será para la misión a realizar por el dron. Está formado por lo que podría ser una sección de una planta industrial. El área está cercada por estanterías y tiene un baño en una de sus esquinas, así como elementos de oficina en la esquina opuesta. Durante la simulación, un conjunto de elementos aparece por las cintas mecánicas que, después de pasar de la primera a la siguiente, acaba en una cinta circular donde un primer brazo manipulador las orientará correctamente cada vez que las detecte con un sensor laser que lleva integrado. Gracias al sistema de mensajería de señales entre cintas, sensores y robots,

estos últimos se encargan de añadir otro tipo de caja más pequeña y gris a la cinta circular de manera sincronizada para que el siguiente robot coloque cada caja pequeña encima de su anterior grande y roja. Cuando va llegando al final del recorrido, otros dos robots se encargan de volver a separar los paquetes apilados, depositando cada uno en una cinta que se los llevaría a otra hipotética sección de la planta.

Este tipo de trabajos proporcionan una mejor y más efectiva solución a los problemas que causan las inspecciones en una planta industrial, que pueden ser el difícil acceso a ciertas áreas y el tiempo y coste de acceder a ellas, ya que en algunos casos se debe apagar la maquinaria para que un operador entre en la zona de trabajo. También proporciona soluciones automáticas e inmediatas al estar conectada al sistema tipo SCADA que lleve integrada la planta.

Aquí la misión del dron es realizar su inspección rutinaria adelantándose a cualquier objeto que se interponga en su trayecto. Entre la inspección del proceso, se pueden encontrar tareas como, comprobación de distancia correcta entre cajas, evaluación de funcionamiento de maquinaria, diagnóstico de fallo mediante localización de cuello de botella, etc.

5.1.3 Escenario 3. Inventario

A este escenario se le suma añade una posibilidad más a evaluar. Uno de las aplicaciones que permiten realizar el avance que suponen los sistemas basados en el comportamiento es la capacidad de decisión en tiempo real para auto carga. La capacidad de decidir el momento idóneo para interrumpir la tarea que se está realizando para ir a la estación de carga, según la carga actual, la tarea que se prevé y la distancia al centro de estación de carga, supone un paso de gigante en la evolución de los robots aéreos autónomos en busca de la autonomía total o inteligencia.

Las tareas de inventario pueden convertirse en crucial para la rentabilidad de una empresa, debido a las pérdidas que pueden ocasionar una mala gestión del stock

y el malgasto que esto supone; y no es tarea fácil en algunos casos en los que se manejen grandes volúmenes de stock.

Este escenario representa una gran sala de almacenaje, repleto de estanterías y pasillos que se cruzan. Durante una misión de inventario, el dron debe recorrer todos los estantes de cada estantería mientras va comprobando que todo esté en el lugar adecuado y contabilizando el producto. Está preparado para realizar serie de simulaciones incrementando el número de drones para validar la correcta coordinación entre ellos, la modificación de ruta para no colisionar y la idoneidad en el momento escogido por el dron para cargarse.

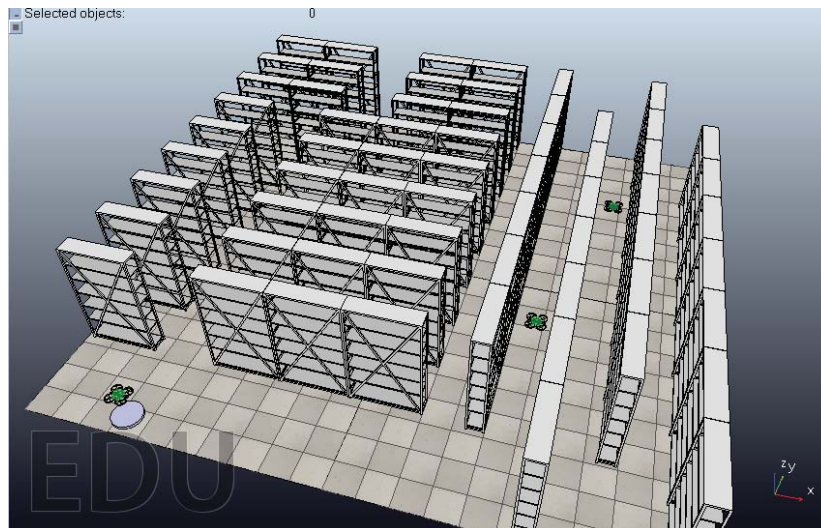


Figura 9. Escenario para simulaciones de tareas de inventario

6.1 Análisis Grado de Adecuación

Como se anunciaba anteriormente, el presente estudio se ha apoyado en varios indicadores de calidad software para plasmar los resultados, cuantificables algunos, e impresiones sobre el grado de adecuación que tiene la herramienta VREP para trabajar como simulador de misiones desarrolladas en Aerostack.

Las pruebas han sido realizadas en un ordenador portátil *hp*, procesador *Intel CORE i5*, de 16 Gb de memoria RAM, con tarjeta gráfica *Intel GMA HD*. Si bien no son malas prestaciones de computación para lo que se pretende, sí que se espera encontrar como obstáculo un ineficiente renderizado de la imagen al simular.

6.1.1 Análisis Técnico

6.1.1.1 Funcionalidad

Uno de los motivos por los que VREP es una de las herramientas más usadas en simulación de robots es precisamente la gran variedad de opciones que te ofrece a la hora de crear tu escenario virtual. Por un lado, destaca la cantidad de motores físicos que ofrece: *Bullet*, *ODE*, *Vortex* y *Newton*. Los motores físicos (*physics engine*) son paquetes software de simulación que proporcionan simulaciones de sistemas físicos, tales como dinámica de cuerpo rígido, dinámica de colisiones, dinámica de fluidos o de cuerpos blandos. Frente a otros simuladores conocidos en el mercado, como Gazebo o ArGOS, supera notablemente en el número de estos motores.

Como segunda función a tener en cuenta, se pueden usar hasta 7 distintos lenguajes de programación, siendo el de defecto LUA. Ya sea mediante scripts

embebidos adjuntos a cada objeto o robot, plugins, nodos de ROS o por medio de programas externos conectados mediante *RemoteAPI*. Haciendo uso del API normal, se puede programar en C o LUA, pero mediante API remotos, puede extenderse su uso a C++, Python, Java, Matlab or Octave. Además, posee una librería propia con más de 200 funciones. La mayoría de competidores ofrecen dos o tres alternativas de programar entre C++, Lua, Python y Matlab. Gazebo da la posibilidad de programar en C++ y nodos de ROS, en ARGoS los robots pueden ser programados mediante scripts en Lua o C++, mientras que RobotDK en Python, C# o Matlab.

Es posible crear figuras, desde un básico cubo o disco, hasta modelos robóticos complejísimos, ofreciendo un sistema jerarquizado de elementos que permiten ‘unir’ estos para que hereden movimientos y demás cualidades físicas. Otros simuladores no ofrecen esta funcionalidad, ya que importan modelos robóticos como unitarios sin poder manipular cada subelemento de manera individual. También, estas formas pueden ser modificadas, dando así un paso más hacia una simulación realista ya que puede emular actuadores cortando, embutando o separando una misma forma. La complejidad de los modelos por defectos de la herramienta es un factor que puede ser positivo, si se busca simulaciones de alta precisión, o negativo, si el problema está enfocado a simular tareas computacionalmente complejas.

Se puede concluir que, bajo el análisis de la funcionalidad, VREP destaca por encima de sus competidores en el mercado. Además, en la universidad no se tiene mucho margen de cambio para elegir zona o elementos con los que interaccionar a la hora de crear un escenario para las misiones que se desarrollan y que pretenden ser probadas en entornos reales. Por lo tanto, es una buena opción tener una herramienta de simulación que te dé la seguridad de que será posible crear un entorno muy similar al real, por particular que éste pueda ser. La tabla comparativa siguiente resume lo anterior.

Software	Gazebo	VREP	ARGos
Multiplataforma	MacOS/Linux/Windows	MacOS/Linux/Windows	MacOS/Linux
Motores Físicos	ODE	Bullet/ODE/ Vortex/Newton	Motores personalizados muy limitados
Manipulación Formas/Modelos			
Programación	PlugIns/ROS	'ROSNodesPlugIns/Remc	Scripts
Lenguajes	C++/ROS	C/C++/Lua/Python/ Java/Matlab/Octave	Lua/C++
Exportabilidad	Sí, XML	NO	Sí, XML

Tabla 1. Comparativa con principales competidores

6.1.1.2 Confiabilidad

Para medir la confiabilidad de VREP, el estudio se ha basado más bien en lo que podría denominarse *integrabilidad*, siendo ésta la facilidad con la que se configurar y se comunica la herramienta con los distintos componentes software, en este caso ROS y Aerostack. ROS es un framework que actúa como sistema operativo distribuido orientado a la robótica para gestionar y comunicar varios equipos en red de una manera sencilla. Está basado en una arquitectura de grafos donde el procesamiento toma lugar en los nodos que pueden recibir, mandar y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores, entre otros.

La herramienta VREP ofrece facilidades para actuar como un nodo más con la que otros nodos pueden actuar por medio de servicios, y sistema publicador/subscriptor. De hecho, es posible programar los elementos de un escenario mediante un plugin de ROS con *ROSNodes*. Puesto que Aerostack está ejecutado sobre ROS, las comunicaciones son llevada a cabo por ROS, lo que facilita el trabajo en gran medida.

Simplemente, copiando el fichero correspondiente del plugin de ROS en la carpeta VREP de Aerostack permite la comunicación entre los scripts embebidos de VREP y Aerostack a través de ROS.

6.1.1.3 Eficiencia

Aunque la potencia y capacidad de la herramienta es bastante alta, ésta se puede ver limitada por la también alta exigencia de procesamiento para las prestaciones del ordenador en el que se ejecuta, que son:

- Procesador Intel Core i5-8250 (4 núcleos, 6 MB caché, 1.6/3.4 GHz)
- Memoria RAM 16 GB DDR4
- Disco SSD 256 MB
- Tarjeta Gráfica Intel HD Graphics 620
- Ubuntu LTS 18.04

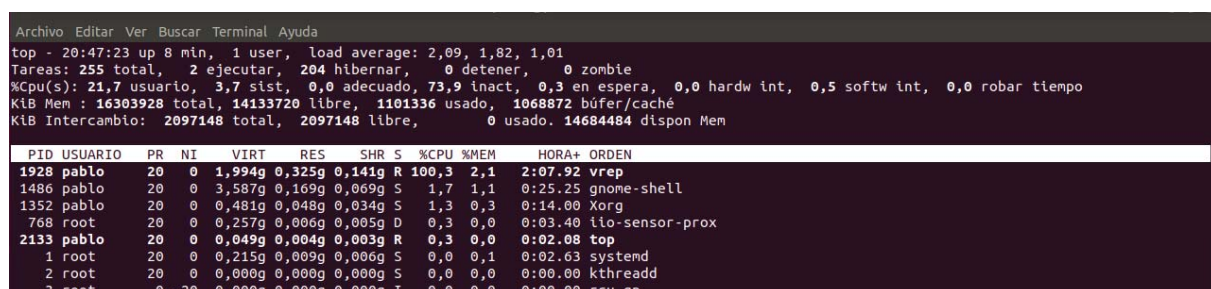
Esto se puede ver claramente en el caso gráfico en el cual, con la tarjeta Intel usada no es capaz de mostrar gráficamente la simulación de tal manera que no se entrecorte la renderización. Por esto, no destaca en eficiencia ya que requiere una tarjeta gráfica de gran potencia, además que, incluso en las más simples simulaciones, ocupa cerca del 100% de la CPU.

Comprobar visualmente lo que va ocurriendo en una simulación puede ser de gran ayuda. Sin embargo, no es algo indispensable para la mayoría de las utilidades que se le pueden dar desde Aerostack, ya que el éxito o fracaso de una misión podría ser medido por muchos otros indicadores aparte del factor visual.

Se somete a una prueba haciendo uso del escenario 3. A modo de recordatorio, este escenario estaba pensado para probar misiones con tareas de inspección de inventario en un almacén

Durante la simulación una serie de drones del mismo tipo, *AR Drone*, realiza el recorrido programado realizando conteos de cada producto que encuentra, con la posibilidad de que trayectorias de dos drones se crucen. Se realiza la misma simulación, bajo similares condiciones, a excepción de que se irán incrementando el número de drones para realizar la tarea, la cual será idéntica para todos ellos.

Mediante el uso del comando *top* en la consola Linux, podemos mostrar una lista con todos los procesos que estén compartiendo la CPU. Esta lista, como se aprecia en la Figura 10, muestra un resumen del estado del sistema. Son de especial interés la segunda fila, que indica el número de tareas desglosadas en su estado: ejecutándose, hibernando, o huérfanas (zombie, procesos con proceso padre muerto). Las siguientes dos filas muestran el uso de las memorias RAM y la de intercambio. La memoria de intercambio, o swap, es una porción de la memoria ROM que el sistema se ve obligado a usar cuando la RAM se completa. Finalmente, aparece una lista de cada uno de los procesos con la cantidad de memoria y CPU usados.



```

Archivo Editar Ver Buscar Terminal Ayuda
top - 20:47:23 up 8 min, 1 user, load average: 2,09, 1,82, 1,01
Tareas: 255 total, 2 ejecutar, 204 hibernar, 0 detener, 0 zombie
%Cpu(s): 21,7 usuario, 3,7 sist, 0,0 adecuado, 73,9 inact, 0,3 en espera, 0,0 hardw int, 0,5 softw int, 0,0 robar tiempo
KiB Mem : 16303928 total, 14133720 libre, 1101336 usado, 1068872 búfer/cache
KiB Intercambio: 2097148 total, 2097148 libre, 0 usado, 14684484 dispon Mem

```

PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	HORA+	ORDEN
1928	pablo	20	0	1,994g	0,325g	0,141g	R	100,3	2,1	2:07.92	vrep
1486	pablo	20	0	3,587g	0,169g	0,069g	S	1,7	1,1	0:25.25	gnome-shell
1352	pablo	20	0	0,481g	0,048g	0,034g	S	1,3	0,3	0:14.00	Xorg
768	root	20	0	0,257g	0,006g	0,005g	D	0,3	0,0	0:03.40	ilo-sensor-prox
2133	pablo	20	0	0,049g	0,004g	0,003g	R	0,3	0,0	0:02.08	top
1	root	20	0	0,215g	0,009g	0,006g	S	0,0	0,1	0:02.63	systemd
2	root	20	0	0,000g	0,000g	0,000g	S	0,0	0,0	0:00.00	kthreadd
3	root	0	-20	0,000g	0,000g	0,000g	I	0,0	0,0	0:00.00	rcu_gp

Figura 10. Salida de comando 'top' en consola Ubuntu

Se han simulado 2 escenarios bajo un total de 5 situaciones distintas, recopilando la información obtenida del comando *top* en cada una de ellas. Se fue aumentando el número de drones en el escenario dedicado al inventario de almacén hasta un máximo de 4 drones. Finalmente, se hace la misma prueba para el escenario de la planta industrial ya que, como se puede intuir por la lentitud de renderización, es la que más carga computacional tiene. En la gráficas de la Figuras 11 y 12 se muestran los resultados.

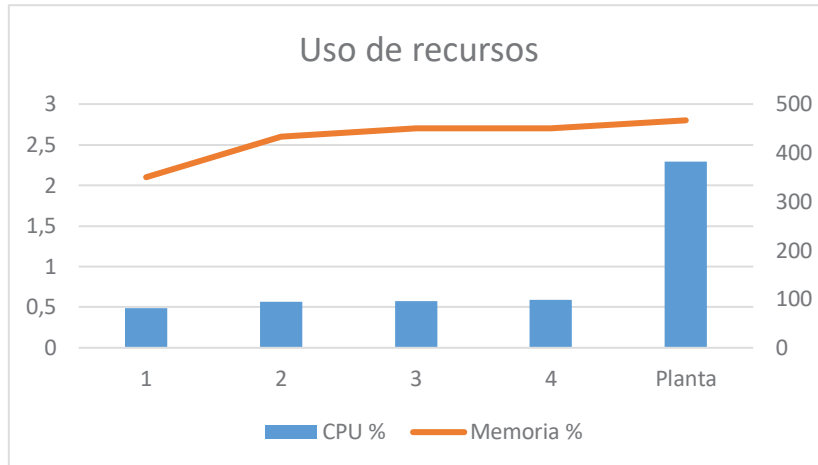


Figura 11. Utilización de recursos para distinto número de robots

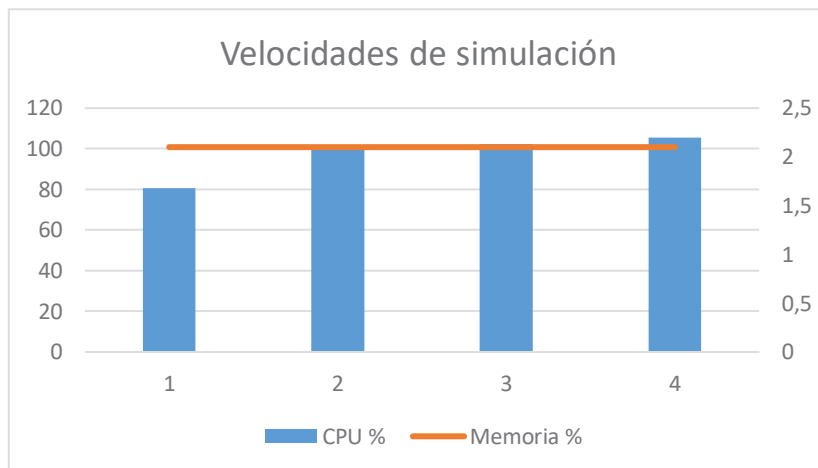


Figura 12. Utilización de recursos para distintas velocidades de simulación

Con el objetivo de encontrar los límites de procesamiento y renderizado a partir de los cuales la herramienta dejaría de ser funcional, se ha continuado con estas pruebas, comprobando ahora la evolución de la eficiencia en función al número de drones en juego, medido en tiempo de ejecución. Esto es, sobre el mismo escenario y bajo una tarea concreta y predefinida, se realizan una serie de simulaciones en las que se duplica el número de drones con respecto a la simulación anterior. Los drones deben realizar conjuntamente la tarea, por lo que, cuanto mayor es el número de drones participantes más simple es la tarea de cada uno. Para este caso concreto, se ha comenzado con 4 drones, para pasar a 8 y hasta 16. Los primeros 4 drones realizan

una pasada por todos los pasillos que forman los armarios del escenario. Como se decía, para los siguientes casos, el recorrido de cada dron se reduce. El medidor principal de esta prueba es el tiempo, se cuenta el tiempo requerido para la tarea para los tres casos. Los resultados de esta prueba quedan reflejados en la *Figura 13*.

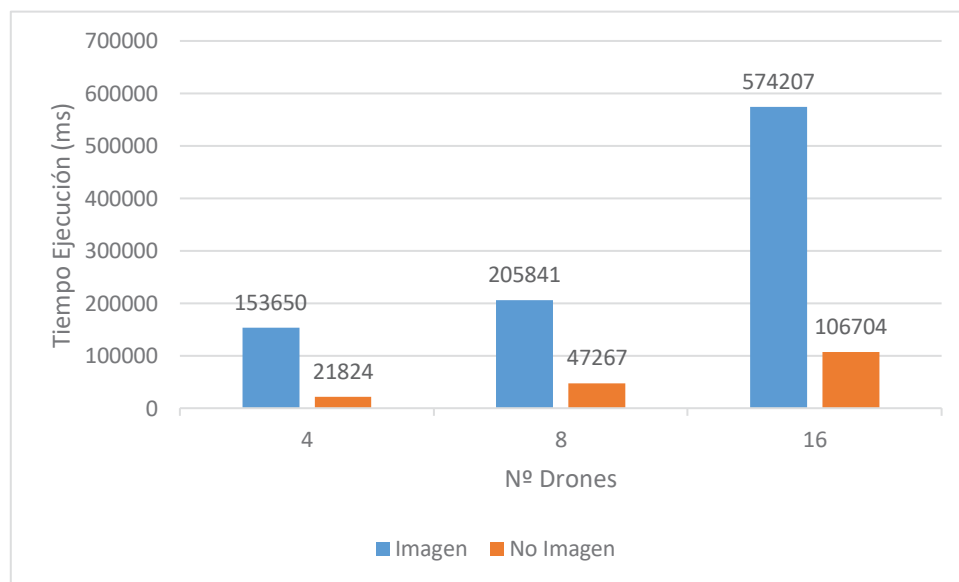


Figura 13. Resultados prueba tiempo para 4, 8 y 16 drones

La prueba finaliza testeando 16 drones debido a que, para ese número, la simulación se hace excesivamente lenta. De hecho, con procesamiento de imagen de simulación, se llega a ralentizar hasta diez veces el tiempo real. Tras realizar varios intentos por acelerar la simulación, (existen varias opciones: un mayor número de procesamientos del *mainscript* por unidad de tiempo, menos rendizados por unidad de tiempo o un factor de multiplicación para el caso de simulaciones en tiempo real), no se obtiene variación perceptible en el tiempo de ejecución de la misión. Cuando la misión es simulada en tiempo real, el factor de multiplicación se topa con alguna otra limitación y apenas provoca aceleración; para el caso en el que la opción de tiempo real aparece desactivada, existe la posibilidad de reducir la precisión de la simulación

en aras de conseguir velocidad, pero, debido quizá a comportamientos inesperados en el movimiento de los drones, la realización de la tarea se hace incluso más lenta.

El factor que verdaderamente reduce el tiempo de simulación de la ejecución es la posibilidad de poder desactivar la renderización. Mediante la llamada a una función es posible prescindir de la imagen 3D y mejorar bastante los tiempos de simulación, llegando a reducirse en más de 85%.

Se puede sacar en claro que, si bien el mero hecho de aumentar el número de agentes no incrementa demasiado el coste computacional y el uso de recursos computacionales, sí que lo hace la renderización de la imagen. Es decir, para los primeros cuatro casos, el aumento de drones apenas afecta; sin embargo, cuando aumentamos en mayor medida el número de drones, hasta dieciséis, la simulación se ralentiza bastante. Para estos casos, la renderización se ve bastante afectada, ya que la tarjeta gráfica de trabajo no es capaz de procesar la simulación correctamente.

Sí que es importante destacar que, dada las limitadas opciones que tiene la herramienta para el aumento de la velocidad de simulación, ésta no serviría de gran ayuda para importantes y pesados aprendizajes.

6.1.2 Análisis cualitativo

6.1.2.1 Usabilidad

Este indicador es fácilmente analizado, puesto que debido a la suficiente y clara documentación que ofrece la web oficial junto a la interfaz gráfica de usuario que ésta posee, la herramienta VREP consigue una excepcional usabilidad para el usuario. Por un lado, la botonería que posee y las barras de opciones de la herramienta son bastante intuitivas, lo que hace bastante sencillo llegar a crear un modelo por uno mismo sin necesidad de consultar la documentación. Se puede complicar algún proceso al encontrarnos con tantas opciones, especialmente propiedades dinámicas, pero todo está documentado; además, conseguir simpleza eliminando estas opciones de las propiedades sería hacerlo a costa de su funcionalidad.

Es por esto que es una de más comúnmente usadas herramientas de simulación para enseñanza en las universidades.

6.1.2.2 Mantenibilidad

Debido a su extendido uso comercial y educacional, es un software con bastante soporte y mantenimiento por parte de la empresa desarrolladora, Coppelia Robotics. Tiene varias versiones anteriores, y cada nueva versión está probada de tal manera que no causa problemas de compatibilidades con los sistemas operativos del momento. Además de esto, cabe destacar que uno de los motivos principales por los que el presente trabajo tomó este rumbo, cuando inicialmente se pretendía llegar a ejecutar misiones de Aerostack en VREP y no sólo la creación de los entornos, viene derivado de la falta de compatibilidad actual entre VREP y Aerostack, con ROS como middleware, cuando es ejecutado sobre la última versión LTS de Ubuntu, la 18.04.

6.1.2.3 Portabilidad

Este indicador puede venir determinado por su cualidad multiplataforma. La web oficial anuncia que está preparado para ser ejecutado tanto en distribuciones Linux, como Windows y MacOS. Sin embargo, y de nuevo bajo la experiencia personal, es sabido que se pueden experimentar bastantes problemas con la herramienta cuando se ejecuta en Windows, por eso siempre es aconsejable hacerlo sobre Linux. Por otro lado, en cuanto a la accesibilidad a la misma, *coppeliarobotics* ofrece una versión educativa para estudiantes con prácticamente toda la funcionalidad original, lo que la convierte en una opción ideal para este tipo de cometido.

CONCLUSIONES

Una vez llegado a este punto, el lector deberá tener un conocimiento general de la robótica aérea, de los drones aéreos autónomos y de los principales paradigmas que consiguen ofrecer esta posibilidad de dotarles con cada vez más inteligencia. Asimismo, desde el grupo de investigación que trabaja diariamente con Aerostack, se podrá realizar un estudio exhaustivo y comparativo con las demás posibilidades existentes para simular las misiones creadas o realizar procesos de aprendizaje, todo esto con la información provista aquí. Además de esto, se deja en posesión del grupo de investigación, los ficheros *.txt* de los entornos creados para que hagan uso de ellos en el momento en el que se consigan comunicar ambos software sobre la última versión Ubuntu.

A partir de este documento, se puede estimar la magnitud de importancia que va adquiriendo el uso de los drones autónomos en tantas aplicaciones, de casi todos los ámbitos posibles, ya sea militar, como civil y cada vez más incluso para servicios y como uso personal. Permiten reducir costes, en tiempo y dinero, así como riesgos para muchos trabajadores del sector primario y secundario, sumándole el aliciente de poder interactuar con ellos sin que suponga un peligro para el trabajador.

Enfocándose ahora sobre la herramienta VREP, ha quedado demostrado que se trata de una de las alternativas existentes más atractivas en el mercado de la simulación software de robots. Con un equipo de prestaciones medias y una buena tarjeta gráfica, ofrece infinidad de posibilidades para numerosas aplicaciones.

Concretamente, para el caso del uso como simulador de misiones desarrolladas en Aerostack, es la herramienta idónea. Esto es debido, principalmente, a su facilidad de acceso mediante la versión gratuita de estudiante y al hecho de poder ser interpretada como un nodo ROS, gracias a la API que ofrece.

Sí que se han encontrado ciertas limitaciones que podrían registrarse como líneas futuras de investigación y desarrollo, como por ejemplo la inflexibilidad en el control de la velocidad de simulación, lo que dificulta un examen exhaustivo de estos incrementos y su utilidad para casos como el aprendizaje por refuerzo acelerado. También podría seguirse por una línea de continuidad hacia la integración total de ambas herramientas, de tal manera que se consiguiese aprovechar los beneficios del simulador, aportando a través de Aerostack mayores posibilidades.

REFERENCIAS

1. B. B. Edin, L. Beccai, L. Ascari, S. Roccella, J.J. Cabibihan, M.C. Carrozza: A bio-inspired approach for the design and characterization of a tactile sensory system for a cybernetic prosthetic hand, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2006) pp. 1354–1358
2. B. Siciliano, O. Khatib, Handbook of Robotics (Springer, 2nd Edition)
3. Burema et al.: Aerial farm robot system for crop dusting, planting, fertilizing and other field jobs. United States Patent (2014)
4. C. Arkin, Ronald – Behavior-Based Robotics. The MIT Press, Massachusetts (1998)
5. C. H. Lin, T.W. Erickson, J.A. Fishel, N. Wettels, G.E. Loeb: Signal processing and fabrication of a biomimetic tactile sensor array with thermal, force and microvibration modalities, Proc. IEEE Int. Conf. Robotics Biomim. (ROBIO) (2009) pp. 129–134
6. D. Um, V. Lumelsky: Fault tolerance via component redundancy for a modularized sensitive skin, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (1999) pp. 722–727
7. E. Cheung, V. Lumelsky: A sensitive skin system for motion control of robot arm manipulators, Robotics Auton. Syst. 10(1), 9–32 (1992)
8. F. Michaud, M. J. Mataric: Learning from History for Behavior-Based Mobile Robots in Non-Stationary Conditions. Machine Learning, 31(1-3), pp 141-167 (1998)
9. F. Michaud, M. T. Vu: Managing robot autonomy and interactivity using motives and visual communication. (1999)
10. G. Darivianakis, K. Alexis, M. Burri, R. Siegwart: Hybrid predictive control for aerial robotic physical interaction towards inspection operations, Proc. IEEE Int. Conf. Robotics Autom. (ICRA) (2014)
11. G. L. McAdoo. UAV configurations and battery augmentation for UAV internal combustion engines, and associated systems and methods. United States Patent Application Publication (2017)
12. G. N. Saridis: Intelligent robotic control, IEEE Trans. Autom. Control AC-28(5), 547–557 (1983)

13. G. Sagnac: L'ether lumineux demontre par l'effect du vent relatif d'ether dans un interferometre en rotation uniforme, C. R. Acad. Sci. Paris 157, 708-710 (1913)
14. Hada, Y., et al.: Information acquisition using intelligent sensor nodes and an autonomous blimp. In: Proc. SICE Annual Conference, pp. 988–991 (2008)
15. <https://www.iso.org/standard/64764.html>
16. J. Blythe, W. Scott Reilly: Integrating Reactive and Deliberative Planning for Agents. School of Computer Science Carnegie Mellon University, Pittsburgh, PA (1993)
17. J. Busset, F. Perrodin, P. Wellig, B. Ott, K. Heutschi, T. Rühl, T. Nussbaumer: Detection and tracking of rones using advances acoustic cameras. *SPIE Security + Defense* (2015)
18. L. C. Fernandes, J. R. Souza, G. Pessim, P. Y. Shinzato, D. O. Sales, V. Grassi Jr., K. R. L. J. Branco, F. S. Osorio, and D. F. Wolf. CaRINA intelligent robotic car: Architectural design and implementations. *Journal of Systems Architecture*, 2013.
19. L. Marconi, R. Naldi, L. Gentili: Modelling and control of a flying robot interacting with the environment, *Automatica* 47(12), 2571–2583 (2011)
20. Lighter-Than-Air UAVs for Surveillance and Environmental Monitoring Michael Gerke, Ulrich Borgolte, Ivan Mas'ar, František Jelenciak, Pavol Bahník, and Naef Al-Rashedi
21. M. Nicolescu, M.J. Matarić: Learning and interacting in human-robot domains, *IEEE Trans. Syst. Man Cybern.* 31(5), 419–430 (2001)
22. M. Schoppers: Universal plans for reactive robots in unpredictable domains, *Proc. Int. Jt. Conf. Artif. Intell.* (1987) pp. 1039–1046
23. Meurer, H., et al.: An Emerging Remote Sensing Technology and its Potential Impact on Mine Action. In: *Proc. International Symposium Humanitarian Demining*, pp. 66–70 (2010)
24. P. Bergveld: Development and application of chemical sensors in liquids. In: *Sensors and Sensory Systems for Advanced Robots*, NATO ASI Series, Vol. 43, ed. by P. Dario (Springer, Berlin, Heidelberg (1988) pp. 397–414
25. P. Bonasso, R.J. Firby, E. Gat, D. Kortenkamp, D.P. Miller, M.G. Slack: Experiences with an architecture for intelligent reactive agents, *Proc. Int. Jt. Conf. Artif. Intell.* (1995)
26. P. Maes, R. A. Brooks: Learning to coordinate behaviors, *Proc. 8th Natl. Conf. Artif. Intell., AAAI* (1990) pp. 796-802
27. R. A. Brooks: Intelligence without representation, *Artif. Intell.* 47, 139–159 (1991)
28. R. C. Arkin: *Behavior-Based Robotics* (MIT Press, Cambridge 1998)

29. R. Hartley, A. Zisserman: Multiple View Geometry in Computer Vision (Cambridge Univ. Press, Cambridge 2000)
30. Rabl, A., Salner, P., Büchi, L., Wrona, J., Mühlbacher-Karrer, S., & Brandstötter, M. (2018). Implementation of a Capacitive Proximity Sensor System for a Fully Maneuverable Modular Mobile Robot to Evade Humans. In *Austrian Robotics Workshop 2018* (p. 17).
31. S. Guitron, A. Guha, S. Li & D. Rus (2017, May). Autonomous locomotion of a miniature, untethered origami robot using hall effect sensor-based magnetic localization. In 2017 IEEE International Conference on Robotics and Automation (ICRA)(pp. 4807-4813), IEEE.
32. S. Mahadevan, J. Connell: Automatic programming of behavior-based robots using reinforcement learning, *Artif. Intell.* 55, 311–365 (1992)
33. S. Walker, K. Loewke, M. Fischer, C. Liu, J.K. Salisbury: An optical fiber proximity sensor for haptic exploration, *Proc. IEEE Int. Conf. Robotics Autom. (ICRA)* (2007) pp. 473–478
34. T. J. Prescott, M.J. Pearson, B. Mitchinson, J.C. Sullivan, A. Pipe: Whisking with robots: from rat vibrissae to biomimetic technology for active touch, *IEEE Robotics Autom. Mag.* **16**(3), 42–50 (2009)
35. T. Martínez-Marín, T. Duckett: Fast Reinforcement Learning for Vision-guided Mobile Robots. *Proc. IEEE Int. Conf. Robotics Autom* (2015)
36. V. Crespi, A. Galstyan, K. Lerman: Top-down vs down-up methodologies in multi-agent system design. *Auton Robot* (2008)
37. Z. Lin: UAV for mapping – low altitude photogrammetric survey, *Proc. 21st ISPRS Congr. Techn. Commis. I, Beijing* (2008) pp. 1183–1186