Actas ROBOT 2011. 28-29 de Noviembre de 2011. Sevilla (España)

Algoritmo Minimax aplicado a vigilancia con robots móviles

S. Vera, A. Jiménez-González, G. Heredia y A. Ollero

Abstract— **En este artículo se aplica la teoría de juegos para planificar el movimiento de un robot en tareas de vigilancia. Se propone el uso de conocimiento heurístico y un algoritmo Minimax para reducir la complejidad computacional involucrada en la implementación de los métodos de la teoría de juegos, que requieren la búsqueda en un árbol de estados. El artículo presenta el algoritmo y su comparación mediante simulación con el conocido método del "líder-seguidor". El artículo incluye también la experimentación con un robot Pioneer en tareas de vigilancia**

I. INTRODUCCIÓN

El desarrollo de estrategias de vigilancia basadas en el empleo de robots móviles es un tema que ha sido considerado de interés en los últimos años [1][2], tanto en vigilancia de perímetros [3][4], como en vigilancia de áreas. Una estrategia de vigilancia consiste en aplicar una serie de robots móviles a través de una determinada área con el fin de evitar cualquier intrusión en dicha área. En diversas publicaciones se han estudiado tanto estrategias deterministas [4], que dependiendo de su complejidad, pueden ser más o menos vulnerables en lo que respecta a la detección de un intruso, como estrategias puramente aleatorias, normalmente más robustas, especialmente si el intruso está observando la estrategia de movimientos del vigilante. No obstante, las estrategias aleatorias no siempre son eficientes en lo que a expectativas y prioridades del vigilante e intruso se refiere.

Los problemas de vigilancia con robots móviles se han estudiado también en el marco de la teoría de juegos [6]. La idea consiste en que el vigilante y el intruso realizan un juego de estrategia cuya salida vendrá condicionada por las combinaciones de sus movimientos entre las distintas celdas. En [5][7][8] se propone una solución, basada en el concepto de equilibrio líder-seguidor ("leader – follower"). Este concepto consiste en un juego de dos jugadores adversarios con información incompleta desde el punto de vista de cada uno de ellos. En este juego se supone que el intruso juega con cierta ventaja, encontrándose en una posición tal que le permite observar la estrategia del vigilante. El concepto de equilibrio de este tipo de juegos consiste en maximizar los recursos del vigilante, teniendo en cuenta que está siendo observado. Sin embargo, este algoritmo precisa de un coste computacional elevado. Los métodos descritos en [7][8] se basan en una programación bilineal, necesitando unos tiempos de computación bastantes altos y de carácter exponencial con el aumento de celdas del grafo de representación.

La alternativa que se propone en este artículo es abordar el mismo marco de juego, siguiendo la misma definición anterior, pero utilizando el algoritmo Minimax para resolverlo. Este algoritmo realiza una valoración de todas las posibles estrategias mediante un árbol de cobertura, desarrollado hasta una determinada cota.

La aplicación eficiente del algoritmo Minimax depende de la utilización de una heurística adecuada al problema. La heurística que se ha desarrollado en el método propuesto en este artículo se basa, por una parte, en un sistema de ganancias proporcionales al valor del objetivo y por otra en la optimización del camino a recorrer entre el vigilante y el posible objetivo, eligiendo la ruta más corta hasta el objetivo evitando que el vigilante vuelva a tomar caminos ya vigilados recientemente.

La implementación del algoritmo propuesto está pensada para robots móviles como el de la Figura 1, dotado de sensores de navegación y sensores de percepción del entorno, tales como cámaras o sensores de distancia, que detecten la invasión de un intruso. El área a vigilar puede ser cualquier planta de un edificio o almacén, que sea estructurado en celdas. El robot, en cada celda, realizará un barrido girando sobre su centro, y si no detectase ningún intruso, ejecuta el algoritmo Minimax, para elegir la celda adyacente a la que moverse para continuar con la vigilancia.

Figura 1. Robot móvil Pioneer utilizado en los experimentos

El resto del artículo se organiza de la siguiente forma: en la sección II se presenta el algoritmo Minimax utilizado.

Santiago Vera, Adrián Jiménez-González, Guillermo Heredia y Aníbal Ollero pertenecen al Grupo de Robótica, Visión y Control, Universidad de Sevilla, Camino de los Descubrimientos, s/n, 41092, Sevilla, España. guiller@cartuja.us.es
Aníbal Ollero también pertenece al Centro Avanzado de Tecnologías Aeroespaciales, CATEC, Parque Tecnológico Aeroespacial de Andalucía Aerópolis; La Rinconada-Sevilla, España. aollero@catec.aero

ISBN 978-84-615-6787-4

La sección III describe las pruebas de simulación y la sección IV presenta los resultados experimentales obtenidos con un robot Pioneer. Por último, la sección V presenta las conclusiones.

II. MINIMAX APLICADO A VIGILANCIA

A. Modelo básico.

El modelo que se propone para la vigilancia de áreas consiste en un sistema compuesto por un vigilante, un posible intruso y algunos objetivos de interés para el intruso, en un determinado entorno. El entorno se modela mediante un conjunto de celdas interconectadas en la que se encuentra el área que se pretende vigilar. Los objetivos se modelan como unas determinadas celdas que tienen un alto valor para el intruso. Se trata de desarrollar una serie de estrategias de vigilancia para robots móviles con el objetivo de conseguir evitar cualquier posible intrusión. La estrategia de vigilancia se encarga de determinar cuál es la próxima celda a visitar. En lo que sigue, se analizan estrategias puras, con un único criterio a seguir y sin aleatoriedad, siendo también posible el desarrollo de estrategias mixtas e introducir un factor de incertidumbre. En el modelado del intruso se considera que sus movimientos están determinados por el valor de los objetivos que pretende alcanzar.

El diseño de este modelo se basa en los usados en [5][7] donde se observa una representación adaptable a cualquier topología que se especializa en sistemas no perimetrales. El modelo se representa por un grafo dirigido $G = (V, A)$, compuesto por un conjunto de nodos V y de arcos A donde

$$V \neq 0 \qquad [1]$$

$$A \subseteq \{(a,b) \in V \times V : a \neq b\} \qquad [2]$$

El conjunto $V = \{c_1, c_2, \ldots. c_n\}$ es un conjunto de nodos o celdas que componen la topología de un determinado recinto y los arcos A definen la interconexión de estos nodos. En la Figura 2 se puede observar un ejemplo sencillo de esta descripción, donde cada nodo o celda está identificada con un número en negrita.

En este modelo el tiempo está discretizado en turnos de manera que el vigilante puede viajar de una celda a otra en un turno, siempre que estas celdas sean adyacentes, es decir, que haya una arista común entre ambas celdas. También se pueden definir caminos de una celda a otra en varios turnos. Posteriormente, en los experimentos, se definirá la duración temporal de un turno, ya que este depende del tipo de robot se utilice. Las celdas representadas con fondo negro, se consideran obstáculos, o simplemente indican que no es posible atravesarlas. En el grafo de adyacencias estas celdas no tienen adyacentes. Cada celda tiene asociado un valor no negativo de interés para el intruso v_i, todas las celdas con un valor mayor a uno son considerados como objetivos T_i, de tal manera que el subconjunto T_i está contenido en V.

0 d0 = 1	**1** d1 = 1	**2** d2 = 1	**3** d3 = 3 v3= 15	**4** d4 = 1
5 d5 = 3 v5= 15		**6** d6 = 1		**7** d7 = 1
8 d8 = 1	**9** d9 = 1	**10** d10 = 1	**11** d11 = 1	**12** d12= 3 v12= 15

Figura 2. Escenario de 13 celdas

El acceso a una determinada celda objetivo por parte de un intruso, requiere un determinado tiempo de penetración, por lo que cada nodo c_i, de T tiene asociado un tiempo de penetración d_i, que modela esta situación. Un intruso, cuando accede a una celda objetivo, debe permanecer en él un total de d_i turnos. Este tiempo es establecido en el modelado de cada problema en concreto, teniendo en cuenta la duración temporal de un turno.

En cada turno el vigilante puede moverse a cualquiera de sus celdas adyacentes, mientras que el intruso tiene dos opciones, puede quedarse fuera del entorno el número de turnos que desee, observando la estrategia del vigilante, o puede entrar directamente en algún nodo objetivo, permaneciendo allí durante d_i turnos sin posibilidad de moverse.

En cada turno la salida del juego puede ser: (1) que el intruso sea detectado, cuando ha entrado en un nodo objetivo c_i en un determinado turno t, y el vigilante visita el nodo c_i en un intervalo de tiempo $[t, t + d_i)$; (2) nodo c_i alcanzado, si el vigilante no llega en ese determinado intervalo; y (3) cuando no es posible que el intruso entre en alguna área.

B. Algoritmo Minimax.

El algoritmo Minimax es una implementación del teorema Minimax de John von Neumann, que se aplica principalmente a la teoría de juegos con un adversario. Minimax es un método de decisión para minimizar la perdida máxima esperada en juegos con adversario y con información perfecta. Este algoritmo se basa en la búsqueda en profundidad de estados que le den la mayor ganancia esperada a una situación determinada del juego, considerando que la existencia de un oponente introduce un factor de incertidumbre. Este algoritmo es muy útil en problemas donde existen limitaciones en el tiempo de respuesta, ya que su tiempo de computación es mucho

menor que otros algoritmos que se basan en resolución por fuerza bruta.

El algoritmo Minimax da origen a una mejora computacional, existiendo una segunda versión mejorada denominada poda alfa-beta. Esta mejora consiste en reducir el árbol de búsqueda utilizando soluciones parciales. El algoritmo, cuando expande una rama de búsqueda, observa si por esa rama va a encontrar una solución mejor que su solución parcial. Si observa que no va a encontrar una solución mejor, desecha dicha rama. Con el algoritmo poda alfa-beta podemos acotar nuestra solución, siendo r el factor de ramificación y p el factor de profundidad, con una complejidad en tiempo de orden $O(r^{\frac{3*p}{4}})$ y en espacio de orden $O(r*p)$.

El algoritmo Minimax se condiciona porque solamente se puede aplicar a juegos bipersonales, en los que los jugadores mueven alternativamente, y en el que la ventaja de un jugador es la desventaja del otro. En estos juegos hay un número finito de estados y decisiones y no interviene el azar. En este artículo se emplea una definición de estado acorde con el problema, desarrollando un mecanismo de expansión de estados que aporte una solución buena y acotada en ramificación y tiempo.

El algoritmo propuesto es de tipo recursivo y consta de los siguientes pasos:

1. Generación del árbol de juego. Se generarán todos los nodos hasta llegar a un estado final o una cota determinada.
2. Cálculo de los valores de la función de utilidad para cada nodo terminal.
3. Calcular el valor de los nodos superiores a partir del valor de los inferiores. Alternativamente se elegirán los valores mínimos y máximos representando los movimientos del jugador y del oponente, de ahí el nombre de Minimax.
4. Elegir la jugada valorando los valores que han llegado al nivel superior.

En nuestro caso el vigilante se define como max y el intruso como min. Un estado completo del juego se define por el número de la celda en la que se encuentra el vigilante, el número de la celda en la que se encuentra el intruso, el tiempo que le resta al intruso para conseguir su objetivo (timeOut), la valoración de una determinada celda en una determinada estrategia y el tipo de jugador (min o max).

El estado inicial del juego es la posición actual del vigilante y la posición del intruso que representa estar fuera del recinto. Un estado será final si el intruso y el vigilante se encuentran en la misma celda, que corresponde a intruso detectado, y si el intruso está en una determinada celda objetivo y su timeOut es cero, el intruso ha conseguido acceder con éxito a un objetivo.

La generación de sucesores y la expansión del árbol de búsqueda se realizan como sigue. La generación de movimientos consiste en la combinación de todos los posibles movimientos del vigilante, que son entre 1 y 4 desplazamientos (dependiendo de la topología), con el número de los objetivos accesibles *T*, que se define entre 1 y *n*. Esta combinación se realiza en el proceso de expansión de un nodo de la siguiente forma: por cada posible movimiento del vigilante se le asocia la búsqueda de un determinado objetivo (para todo los objetivos *T*) y finalmente se le asocia también el caso no atacar ningún objetivo.

C. Heurística

El éxito o el fracaso de este algoritmo Minimax acotado, cuando no se expande todo el árbol de búsqueda, depende del desarrollo de una buena heurística que tenga suficiente información del entorno. Al desarrollar el árbol de búsqueda hasta una determinada cota, y no hasta todos los nodos hojas, se necesita una función de evaluación para un estado intermedio, y esto es posible mediante una heurística que conozca el entorno, la posibilidad de movimientos, el valor de los objetivos y el tiempo de penetración necesario para estos. Combinando todos estos factores se debe encontrar una heurística que sitúe al vigilante en una posición tal que esté cerca de todos los objetivos, en especial de los más importantes, y que, en caso de intrusión, pueda alcanzar al intruso, y en el caso que no sea capaz de alcanzarlo, que la perdida sea del objetivo de menor valor para el vigilante. El desarrollo de nuestra heurística se ha basado en un sistema de ganancias proporcionales al valor del objetivo en cuestión. Estos valores de los objetivos se han ponderado con unos criterios que se comentan a continuación.

La valoración de un determinado estado es un compromiso entre el valor del objetivo y la longitud en números de celdas necesarias para llegar a alcanzar al intruso obteniendo dicho objetivo. También se premian las rutas que sean más cortas para alcanzar dicho propósito, lo que se consigue, ponderando cada estado final de forma inversamente proporcional a la profundidad del árbol en la que se encuentra, que se denomina Nivel. Esto se implementa de la siguiente manera: Si el vigilante se encuentra en la misma celda que el intruso, la ponderación de ese estado es 8 * v_i * Nivel, donde v_i es el valor de ese objetivo. Si el vigilante se encuentra a una celda de alcanzarlo, la ponderación es 6 * v_i * Nivel. Si está a dos celdas, será a 4 * v_i * Nivel, si está a tres celdas, será a 2 * v_i * Nivel, y si está a más, la ponderación es de 1 * v_i * Nivel. Si el intruso se encuentra fuera del recinto, la ponderación es siempre la misma: el valor medio de todos

los v_i.

Otro concepto que se ha introducido como heurística son los mecanismos para evitar que el vigilante vuelva por un camino por el que ya venía. De este modo haremos que la convergencia hacia un nodo de los objetivos sea más rápida, y evite que se quede oscilando entre celdas intermedias en un determinado camino. La implementación de este mecanismo se ha hecho en el método encargado de expandir el árbol de búsqueda. Este método consiste en eliminar de la expansión de un determinado nodo los estados iguales al padre de este nodo, siempre que sea aplicable. Es posible que, por una determinada topología, la única expansión de un nodo sea volver por donde ha venido. En este caso no se ejecutará este mecanismo.

III. RESULTADOS DE SIMULACIÓN

Se ha comparado el algoritmo Minimax con el algoritmo líder-seguidor (leader – follower) implementado con programación bilineal empleado en [7]. Ambos algoritmos se basan en estrategias puras, y definen el planteamiento del problema de forma similar. En este artículo se pretende comparar los tiempos de ejecución de uno y de otro, teniendo en cuenta que los recursos físicos son muy similares, y por lo cual un tiempo de cómputo también muy similar.

Para el cálculo computacional del algoritmo, en el caso de líder-seguidor se ha empleado un sistema SNOPT 7.2 sobre una máquina Pentium R a 3 Ghz con 1 GB de RAM, con un S.O. Linux. En el caso de Minimax se ha utilizado una implementación en C sobre una máquina AMD Phenom x3 con 1 GB de RAM.

En los resultados de simulación de [7] se observa que el tiempo medio de resolución de cada programación bilineal de un ejemplo como el de la Figura 2 es de unos 6 segundos. Con este algoritmo, el tiempo de cómputo está condicionado por el número de celdas totales n, y el tiempo máximo de penetración de los objetivos $\max(d_i)$. En la Tabla 1 se observa como crece el tiempo en relación a los nodos, definiendo una determinada d_i constante para todos.

Nodos	Tiempo (s)
5	0,05
10	4,33
20	80

Tabla 1. Tiempo de cómputo del algoritmo líder-seguidor

Por otro lado, se han realizado múltiples simulaciones con escenarios distintos, de 4, 13 y 30 celdas para el algoritmo Minimax, obteniendo medidas de tiempos muy buenos y no tan dependientes del número de celdas. En las Figuras 2 y 3 se pueden observar los escenarios utilizados. Sobre estos escenarios se han hecho múltiples experimentos modificando el número de objetivos, el valor de cada objetivo y el tiempo de penetración de estos objetivos.

Los órdenes de magnitud del problema del algoritmo Minimax, respecto al tiempo y al espacio de memoria, vienen definidos por el índice de ramificación y por la cota de profundidad y no por el número total de nodos y el tiempo de penetración como en el caso de programación bilineal. Esto es una gran ventaja del algoritmo Minimax, ya que la computación no es dependiente del grafo de representación, lo que hace al algoritmo Minimax mucho más escalable. El parámetro de ramificación tiene rango conocido y definido por las condiciones del problema. Este parámetro se ve afectado por el número de objetivos, que en proporción al número de celdas totales consideradas, es bastante bajo. El otro parámetro que se ha comentado es la cota límite de expansión de búsqueda. Esta cota se ajusta en simulación. En los casos estudiados en las Figuras 2 y 3 con una cota de 3 es suficiente para obtener buenos resultados.

0 d0 = 1	**1** d1 = 1	**2** d2 = 1	**3** d3 = 2 v3= 35	**4** d4 = 1
5 d5 = 2 v5= 15		**6** d6 = 1		**7** d7 = 1
8 d8 = 1	**9** d9= 1	**10** d10 = 1	**11** d11 = 4 v11= 35	**12** d12 = 1
13 d13 = 1	**14** d14 = 1		**15** d15 = 1	**16** d16 = 1
17 d17 = 1	**18** d18 = 1	**19** d19 = 6 v19= 15	**20** d20 = 1	**21** d21 = 1
22 d22 = 1		**23** d23 = 1		**24** d24 = 1
25 d25 = 1	**26** d26 = 1	**27** d27 = 1	**28** d28 = 1	**29** d29 = 1

Figura 3. Escenario de 30 celdas

Uno de las simulaciones que se han desarrollado es el caso que se ilustra en la Figura 2 con dos objetivos (celdas 3 y 12), siendo estos del mismo valor y teniendo el mismo tiempo de penetración. Como resultado se ha observado que el vigilante que parte desde la celda 0, se mueve hacia la celda 3 por el camino más corto y se queda pivotando entre la celda 3 y la celda 4. Obsérvese que existe un camino de longitud 3 entre las celdas 3 y 12, y la celda 4 está en este camino. Por ello, del resultado del análisis se observa que el vigilante se ha posicionado en un punto intermedio entre los dos objetivos, de manera que está al alcance de capturar al

intruso en caso que intentase atacar alguno de los dos objetivos.

Otra simulación que se ha realizado es proponer el mismo caso de la Figura 2 con los mismos objetivos pero con valores distintos. En este caso el objetivo 3 tiene un valor de 15 y un tiempo de penetración de 3, y el objetivo 12 tiene un valor de 35 y un tiempo de penetración de 3. Después de lanzar la simulación se observa como el vigilante parte de la celda 0 y se mueve hacia la celda 3 por el camino más corto y sigue avanzando hacia la celda 12. Después se queda pivotando entre las celdas 7 y 12. El comportamiento es similar al del escenario anterior, sólo que en este caso se queda pivotando cerca del objetivo que tiene más valor de los dos: el objetivo 12.

Se han desarrollado otras simulaciones sobre el escenario de la figura 3, donde se puede observar un comportamiento similar siendo el escenario más grande. Otro comportamiento observado es que, cuando los objetivos se encuentran muy distantes unos de otros, el vigilante, dependiendo de la configuración de los valores v_i de los objetivos, adopta un camino que pasa por todos los objetivos o cerca de ellos.

En el desarrollo de todas estas simulaciones se ha estudiado también la influencia de la cota de profundidad en la búsqueda. En teoría la cota de profundidad del árbol de expansión se define entre 1 y el máximo de los tiempos de penetración de los objetivos. Por otra parte, después de los distintos experimentos realizados, se ha ajustado dicha cota, observándose que, en la práctica, con una cota de 2 es suficiente en caso de 5 celdas, y de 3 en el caso de 30 celdas. Esta conclusión influye mucho en el tiempo de cómputo de este algoritmo, obteniendo un sistema que toma una decisión en muy corto espacio de tiempo.

Los tiempos de ejecución que hemos obtenido del algoritmo Minimax están por debajo de 0,5 segundos, siendo este tiempo invariante tanto en el ejemplo de 13 como en el de 30 celdas.

Para poder comparar los dos algoritmos, se ha definido una métrica denominada TC que representa la cantidad de tiempo de cómputo por nodo.

$$TC = \frac{nodos}{tiempo} \qquad [3]$$

Esta métrica no tiene unas unidades implícitas pero da una idea del índice de tiempo de cómputo en relación al número de nodos. En la Tabla 2 podemos ver el valor de esta métrica dependiendo del número de nodos y del algoritmo en cuestión.

Tabla 2. Comparación de tiempos de cómputo por nodo de los algoritmos líder-seguidor (L-S) y Minimax

Nodos	L - S	Minimax
5	0,01	0,02
10	2,60	0,05
20	4,00	0,025

IV. RESULTADOS EXPERIMENTALES

Para validar los resultados de simulación se han realizado dos experimentos sobre el escenario de la Figura 2. Para realizar estos experimentos se ha utilizado primero el simulador Stage de Player Project, para reproducir el escenario. Este escenario está compuesto por 15 celdas de 4 metros cuadrados. De estas celdas, dos se consideran obstáculos de manera que es imposible pasar a través de ellas. Del resto de celdas se han establecido tres celdas objetivos, que son las celdas 3, 5 y 11 con valores 30, 5 y 20 unidades respectivamente, y con el mismo tiempo de penetración de 4 unidades de tiempo. Teniendo en cuenta que el robot tarda 45 segundos en moverse de una celda a otra, el tiempo de penetración será 3 minutos para cada objetivo.

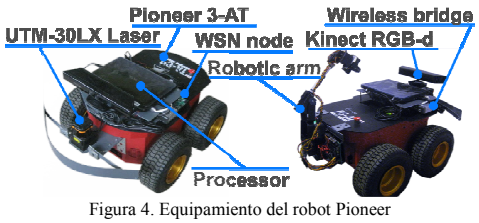

Figura 4. Equipamiento del robot Pioneer

Los experimentos se han llevado a cabo utilizando robots móviles del Testbed Integrado de la Red de Excelencia CONET [11]. Estos robots (Figura 4) están basados en el conocido Pioneer 3-AT de tracción diferencial y movimiento holónomo producidos por MobileRobots, a los que se ha dotado de capacidades sensoriales, de comunicaciones y de procesamiento extras que permiten su utilización en una amplia variedad de experimentos y aplicaciones. Entre sus sensores y actuadores destacan:

- Laser 2D Hokuyo UTM-30LX. Su rango es de 30 m. y 270 grados, tiene una resolución angular de 0.25 grados y una frecuencia de escaneo de 40 Hz.
- Sensor RGB-D (imagen a color más profundidad) Microsoft Kinect. Incluye 4 micrófonos y acelerómetros.
- Sensores de temperatura, humedad y luminosidad integrados en un nodo de Red de Sensores Inalámbricas Telos B.
- Brazo robótico para manipulación de objetos de hasta 300 gr.

Se ha reducido el alcance de la cámara de visión a un metro, ya que las celdas del escenario están definidas de 2 x 2 metros. De esta manera, si hace un barrido circular en el

centro de la celda, tiene suficiente visión para detectar a cualquier intruso. Se puede plantear aumentar el horizonte de visión del robot, con el fin de mejorar la eficacia de la misión.

El algoritmo de decisión se ha implementado en el lenguaje de programación C. La lógica de navegación del robot consiste en viajar de forma autónoma de una celda a otra, usando odometría. Una vez que el robot ha llegado al centro de la celda, tiene programado que realice un giro de 360º sobre sí mismo, comprobando si existe algún intruso en la celda dada. Después de dicha vuelta, si no ha encontrado ningún intruso, selecciona la siguiente celda a la que debe de moverse. En el caso en que después del giro haya detectado un intruso, se parará y dará un mensaje por pantalla.

Se han planteado dos experimentos sobre este mismo escenario. Uno consiste en que el robot parte de la celda número 8 y encuentra un intruso en el objetivo 3, como se observa en la figura 5. El segundo consiste en que el robot parte de la celda 0 y no se encuentra ningún intruso.

Figura 5. Experimento con robot Pioneer en escenario de 13 celdas

Como se ha comentado, en una primera fase del proceso de experimentación se han simulado en Stage estos dos experimentos. A continuación se ha reproducido este mismo escenario en la realidad y se han grabado dos videos: video 1 (http://grvc.us.es/staff/svera/Videos/Video1.wmv) sobre captura de un intruso, y el vídeo 2 (http://grvc.us.es/staff/svera/Videos/Video2.wmv) sobre el proceso de vigilancia.

En el video 1 se puede observar como el robot sale de la celda 8 y se dirige hacia la zona donde se concentran los objetivos de mayor valor (celda 3). En este proceso de navegación, el robot realiza un giro de 360º en cada celda, imitando a cualquier sistema de vigilancia. También se puede observar que entre los dos posibles caminos que existen entre las celdas 8 y 11, elige el camino que pasa por la celda objetivo 5.

En dicho video se puede observar que en el momento en el que el robot se encuentra en la celda 1, entra un ladrón en la celda 3, pero como el robot se encuentra a cuatro metros de distancia, no se ha percatado aún. El robot sigue su estrategia de vigilancia según las prioridades que tienen establecidas. Cuando el robot llega a la celda 11, es capaz de ver al intruso. Justo después de dar la vuelta completa sobre sí mismo en la celda 11, el robot comunica que ha encontrado un intruso a través de un mensaje por pantalla y se para, dándose por terminado el experimento de forma satisfactoria.

En el video 2 se puede observar como el robot sale de la celda 0, y se dirige hacia la zona donde se concentran los objetivos de mayor valor (celda 3). Es este caso vemos como el robot simplemente elige el camino más corto hacia su objetivo. Una vez alcanzado, el robot se queda pivotando entre las celdas 3 y 2 realizando una vigilancia exhaustiva.

V. CONCLUSIONES

En este artículo se ha presentado una alternativa al uso de algoritmos de juegos tipo líder–seguidor para la vigilancia de áreas empleando robots. Se basa en un algoritmo Minimax acotado que permite obtener tiempos de cómputo mucho menor que los de líder-seguidor, y una buena escalabilidad. La dificultad de este algoritmo está en la necesidad de emplear una buena heurística. Los resultados experimentales avalan el buen funcionamiento del sistema global, consiguiendo que la posibilidad de intrusión sea muy baja, un tiempo de computo por debajo del medio segundo, y una gran escalabilidad del problema, ya que no hay dependencia directa entre el algoritmo y el número de celdas que definen una cierta área escenario.

Del análisis de los resultados experimentales se ha concluido que un parámetro importante para el desarrollo satisfactorio de la misión sería el aumentar el campo de visión o percepción del robot, ya que de esta manera aumentaría considerablemente la eficacia del mismo. Otro aspecto a desarrollar en la lógica de guiado del robot sería el introducir un factor de incertidumbre en la navegación. Este factor consistiría en lanzar órdenes de navegación a una determinada celda aleatoria, y dejar evolucionar al robot, desde esa celda a la de mayor valor, y a través de la ruta que considere conveniente siguiendo el criterio Minimax que tiene programado. De esta manera aumentaría la incertidumbre de un intruso que esté analizando su estrategia.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por la Red de Excelencia CONET de la Comisión Europea (FP7-2007-2-224053), el proyecto de excelencia Robótica Ubicua en entornos urbanos (RURBAN P09-TIC 5121) y el proyecto ROBAIR (Dirección General de Investigación DPI2008-03847).

REFERENCIAS

[1] H.R. Everett, *Robotic security systems*, IEEE Instrumentation & Measurement Magazine, Vol. 6, pp. 30-34, 2003.

[2] D. Di Paola, D. Naso, A. Milella, G. Cicirelli and A. Distante, *Multi-Sensor Surveillance of Indoor Environments by an Autonomous Mobile Robot*, 15th Int. Conf. on Mechatronics and Machine Vision in Practice (M2VIP 2008), Auckland, Australia, December, 2008, pp. 23-28.

[3] A. Marino, L. Parker, G. Antonelli and F. Caccavale, *Behavioral Control for Multi-Robot Perimeter Patrol: A Finite State Automata approach,* Proc. of the 2009 IEEE Int. Conf. on Robotics and Automation (ICRA2009), Kobe, Japan, May, 2009, pp. 831-836.

[4] N. Agmon, S. Kraus, and A. Kaminka, *Multi-Robot Perimeter Patrol in Adversarial Setting*, IEE International Conference on Robotics and Automation Pasadena, CAA USA, May 19-23 2008

[5] F. Amigoni, N. Basilico, N. Gatti, A. Saporiti and S. Troiani, *Moving Game Theoretical Patrolling Strategies from Theory to Practice: An USARSim Simulation*, Proc. of the 2010 IEEE Int. Conf. on Robotics and Automation (ICRA2010), Anchorage, USA, May, 2010, pp. 426-431.

[6] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[7] F. Amigoni, N. Basilico and N.Gatti, *Finding the Optimal Strategies for Robotic Patrolling with Adversaries in Topologically Represented Environments*, Proc. of the 2009 IEEE Int. Conf. on Robotics and Automation (ICRA2009), Kobe, Japan, May, 2009, pp. 819-824.

[8] N. Basilico, N. Gatti and F. Amigoni, *Leader – Follower Strategies for Robotic Patrolling in Environments with Arbitrary Topologies,* in Proc. of the Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS2009), Budapest, Hungary, May,,2009, pp. 57-64.

[9] N.Basilico, N. Gatti, T. Rossi, S. Ceppi and F. Amigoni, *Extending Algorithms for Mobile Robot Patrolling in the Presence of Adversaries to More Realistic Settings*, in Proc. WI-IAT, 2009.

[10] P. Paruchuri, J. Pearce and S. Kraus, *Playing Games for Security: An efficient Exact Algorithm for Solving Bayesian Stackelberg. Games*, in Proc. of the Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS2008), Estoril, Portugal, May, 2008.

[11] A. Jiménez-González, J.R. Martínez-de Dios and A. Ollero, *An integrated testbed for heterogeneous mobile robots and other cooperating objects*, IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS2010), Taipei, Taiwan, October 2010,