Robot 2D Pattern Formation

Máster en Inteligencia Artificial

Universidad Politécnica de Madrid

Pedro Frau

1 Introducción

En un contexto de constante evolución de los sistemas robotizados, nos encontramos ante la necesidad de implementar sistemas que incluyan varios agentes. Uno de los problemas que aparecen está relacionado con la organización espacial de dichos agentes. Por ello, nos resulta de especial importancia la posibilidad de desarrollar sistemas capaces de moverse en un espacio bidimensional en función de unos patrones establecidos.

A lo largo de esta práctica, sentaremos las bases de una posible implementación de estos sistemas tomando como idea principal el sistema *SHAPEBUGS* desarrollado por [Rubinstein, Cornejo y Nagpal, 2014]. En él, observaremos como podemos conseguir que se muevan varios agentes en un espacio 2D en función de unas figuras establecidas y preprogramadas para cada agente. Deberemos prestar especial atención a la comunicación entre agentes ya que ellos deben ser capaces de ubicarse en el espacio y posicionarse correctamente.

Empezaremos en primer lugar haciendo una breve explicaciión del algoritmo y proponiendo una posible implementación en MATLAB. Luego, haremos observaciones sobre los resultados obtenidos mediante la implementación propuesta. Finalmente, presentaremos unas breves conclusiones haciendo una crítica sobre el sistema y abriendo vías de mejora.

2 Implementación

Para este apartado nos centraremos en entender el funcionamiento del algoritmo de organización espacial de los agentes. La idea principal es asignar un coste a cada agente en función de su distancia a un punto. Así pues, se inicializará un agente A_1 en el punto (x_1, y_1) correspondiente a la parte baja-izquierda de la figura. Luego, se inicializarán los demás agentes entorno a A_1.

Así pues, empezaremos por inicializar una matriz compuesta por unos en la zona de la figura, y por ceros en el resto. Una vez hecho eso, seleccionaremos el punto más bajo y más a la izquierda de la figura y le daremos un coste de 2, ese será nuestro primer agente. Después, inicializaremos el resto de agentes, hasta un total de 100 asignando costes mediante el algoritmo de wavefront.

Por ejemplo, en un caso en que dispongamos de una figura cuadrada, podríamos obtener el siguiente resultado:

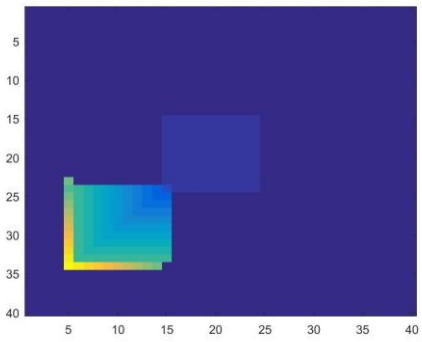

Imagen 1: Ejemplo de inicialización

Como podemos observar tenemos un fondo azul oscuro que correspondería a la zona vacía, un cuadrado en el centro un poco más claro que correspondería a la figura, y un degradado de colores que correspondería a cada agente y su distancia a A_1.

Una vez hecha esta inicialización, debemos saber, que los agentes se irán moviendo en función de su peso. Así pues, primero se moverán los agentes con mayor peso, y les seguirán los de menor peso. Por ello, empezaremos generando una lista de posiciones de los agentes de mayor peso. Esta lista se irá actualizando a cada paso del algoritmo:

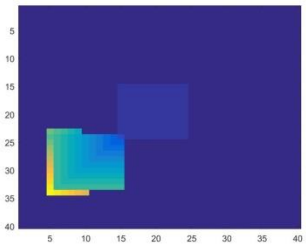

Imagen 2: Ejemplo 1. Movimiento del vector de máximo coste (1)

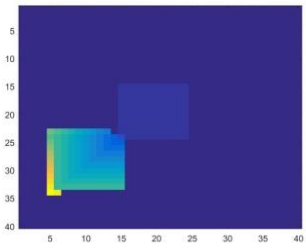

Imagen 2: Ejemplo 1. Movimiento del vector de máximo coste (2)

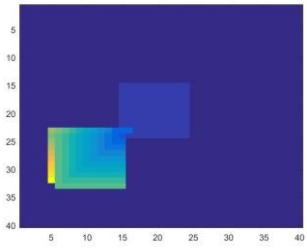

Imagen 4: Ejemplo 1. Movimiento del vector de máximo coste (3)

El vector se seguirá moviendo hasta que rodee al agente con valor 2 dentro de la figura. Una vez logrado eso, a los agentes circundantes se les asignarán valor 2 y pasarán a ser posiciones de referencia.

De la misma manera, una vez que el último agente con peso máximo pase delante de un agente con peso inferior en una unidad, la lista de pesos se actualizará añadiendo los nuevos agentes. De esta manera se irá rellenando la figura:

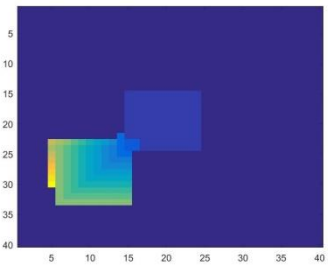

Imagen 5: Ejemplo 1. Generación de la figura (1)

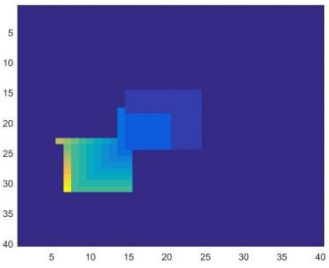

Imagen 3: Ejemplo 1. Generación de la figura (2)

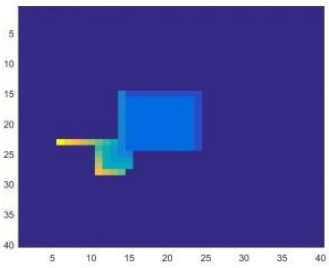

Imagen 7: Ejemplo 1. Generación de la figura (3)

Finalmente, una vez que se ha completado la figura, y todos los agentes están conectados, eliminaremos los agentes sobrantes por conveniencia:

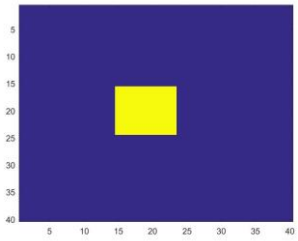

Imagen 4: Ejemplo 1. Resultado del movimiento de los agentes

3 Resultados

Para hacer pruebas, se han generado 4 figuras diferentes, además de la presentada más arriba cuyos resultados podemos observar a continuación:

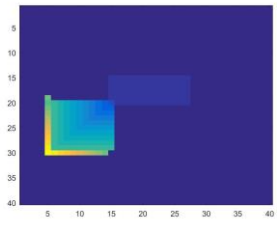

Imagen 9: Ejemplo 2 (1)

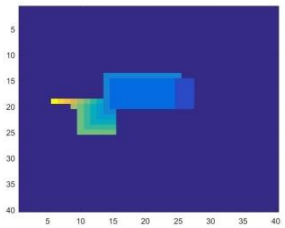

Imagen 10: Ejemplo 2 (2)

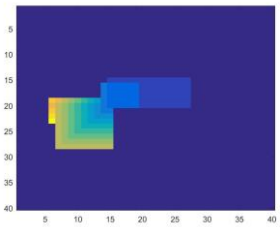

Imagen 11: Ejemplo 2 (3)

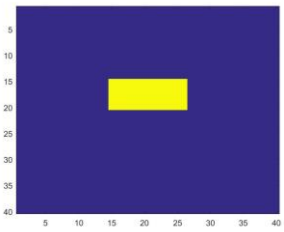

Imagen 12: Ejemplo 2 (4)

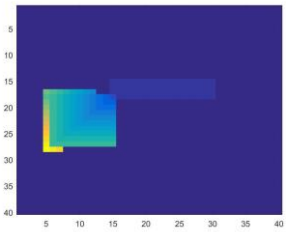

Imagen 13: Ejemplo 3 (1)

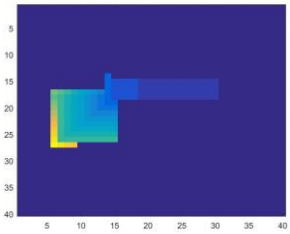

Imagen 14: Ejemplo 3 (2)

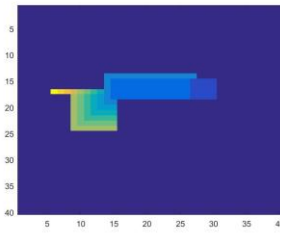

Imagen 15: Ejemplo 3 (3)

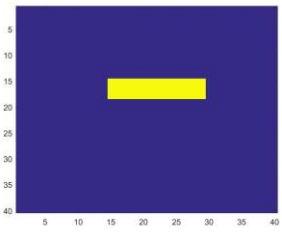

Imagen 16: Ejemplo 3 (4)

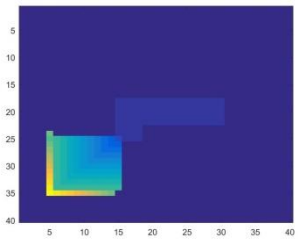

Imagen 17: Ejemplo 4 (1)

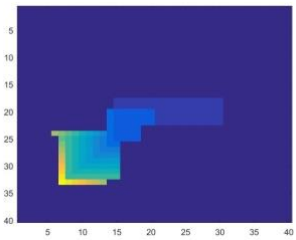

Imagen 18: Ejemplo 4 (2)

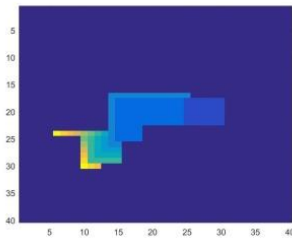

Imagen 19: Ejemplo 4 (3)

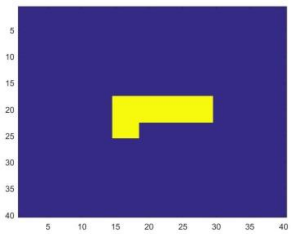

Imagen 20: Ejemplo 4 (4)

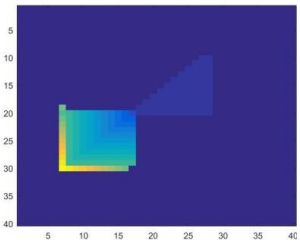

Imagen 21: Ejemplo 5 (1)

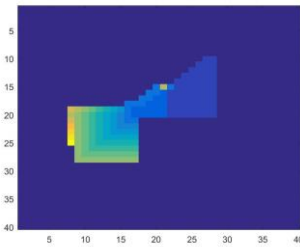

Imagen 22: Ejemplo 5 (2)

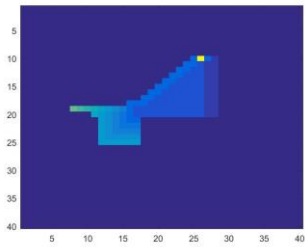

Imagen 23: Ejemplo 5 (3)

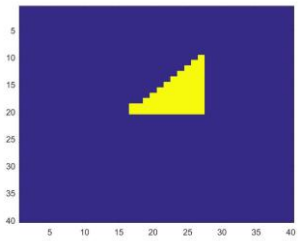

Imagen 24: Ejemplo 5 (4)

En el apéndice se añade el enlace a un vídeo en el que se puede apreciar el

funcionamiento del algoritmo.

4 Extensión

En este apartado se presenta una solución al problema de configuración de varios patrones. Por ello, se genera una matriz en la que aparecen dos figuras que deben ser representadas por los agentes. Para incluir esta mejora, se ha tenido que generar un nuevo líder en la segunda figura, una vez que se ha completado la primera. Estos son los resultados obtenidos:

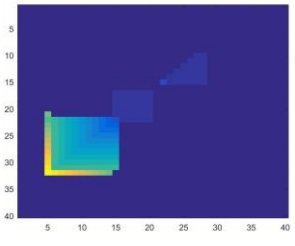

Imagen 5: Extensión (1)

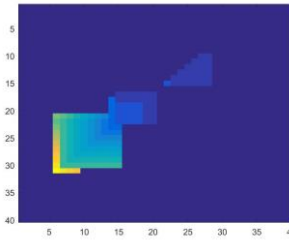

Imagen 26: Extensión (1)

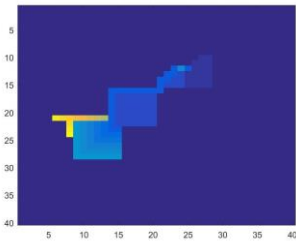

Imagen 27: Extensión (1)

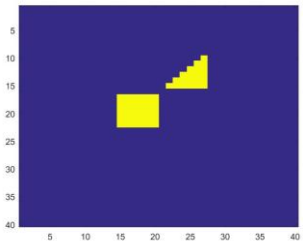

Imagen 28: Extensión (1)

5 Conclusiones

Por lo que hemos podido observar en la realización de esta práctica, la idea propuesta inicialmente por [Rubinstein, Cornejo y Nagpal, 2014] resulta esencialmente fácil de implementar. Aparentemente, lo único que genera complicaciones es el hecho de tener que contemplar numerosas condiciones diferentes que pueden complicar mucho el algoritmo, hacerlo computacionalmente pesado y no exitoso.

A lo largo de la implementación se han ido encontrando muchos errores que no han sido resueltos aún. Estos no se muestran en la presente memoria. Por consecuente, resulta de vital importancia presentar una implementación detallada y bien estructurada de manera a facilitar la localización de estos problemas que presentan una vía de estudio interesante en futuras versiones del algoritmo.

6 Bibliografía

1. **Rubinstein, Cornejo y Nagpal**, 2014, *Programmable self-assembly in a thousand-robot swarm*, Science 345, 795

7 Apéndice

Enlace al vídeo con la presentación de los ejemplos:

https://youtu.be/sog_uqqR3fU