**Clusterix 4.0**

**Software de navegación para enjambres de robots**

**L.E. Ruvalcaba Tamés**

Instituto de Ciencias Nucleares, U.N.A.M.

Laboratorio de Instrumentación Espacial, LINX

CDMX, México

led.ruvalcaba@correo.nucleares.unam.mx ; led.ruvalcaba@gmail.com

**RESUMEN**

La misión lunar COLMENA, desarrollada por el Laboratorio de Instrumentación Espacial (LINX) del Instituto de Ciencias Nucleares (ICN) de la UNAM, consiste en el diseño, construcción y despliegue sobre la superficie lunar de un conjunto de 9 robots autónomos pequeños. El objetivo de esta misión es el de demostrar que un enjambre auto-organizado puede ser utilizado para ensamblar estructuras de interés sobre la superficie expuesta al medio interplanetario de un satélite o asteroide. Se puede así desarrollar una herramienta más de exploración espacial que soporte tanto a robots grandes y complejos como a seres humanos.

**1 DESCRIPCIÓN DEL ALGORITMO DE OPERACIÓN**

Antes de comenzar a hablar del algoritmo en concreto conviene describir un poco el ambiente en el que opera, los componentes con los cuales interactúa y algunos términos recurrentes que serán utilizados a lo largo del presente trabajo.

El elemento mínimo de operación del algoritmo es un robot, un robot de forma individual tiene una serie de características y funcionalidades que se describen a continuación:

Características:

1. Una posición en el espacio tridimensional de trabajo (para motivos de simplicidad el software trabaja en solo dos dimensiones, i.e. altura = 0). Dicha posición es asignada aleatoriamente, ya que depende del lugar donde caiga tras ser eyectado del contenedor.
2. Un ángulo de dirección en el espacio tridimensional de trabajo (para motivos de simplicidad el software trabaja en solo dos dimensiones, i.e. altura = 0). Dicha dirección es asignada aleatoriamente, ya que depende de la dirección hacia donde apunte el frente del robot tras ser eyectado del contenedor.
3. Un radio, es decir, posee dimensiones (en el software se simula usando un circulo de radio configurable).
4. Lóbulo de sensado, función que estima con que intensidad percibe un robot a otros dependiendo de la posición angular del robot lejano respecto a la dirección del robot que se encuentra sensando, es un emulador de la antena que porta cada uno de los robots en los modelos de prueba.
5. Lóbulo de emisión

Funcionalidades:

1. Girar, un robot puede girar θ grados en dirección positiva convencional.
2. Avanzar, un robot puede avanzar una unidad en la dirección de su ángulo de dirección.
3. Sensar, un robot puede percibir a otros robots que estén emitiendo en la misma frecuencia (desde ahora se usará de forma indistinta el término frecuencia o canal).
4. Emitir, un robot puede volverse visible para otros robots que se encuentren sensando en la misma frecuencia en la que el robot está emitiendo.
5. Conectarse, un robot puede unirse a otro robot que se encuentre en un radio de colisión definido, es decir, si están lo suficientemente cerca, ambos forman un conjunto (cluster) y ambos dejan de moverse.
6. Desconectarse, un robot puede interrumpir la conexión que sostiene con otros robots y volver a moverse en la dirección que apunte su algoritmo.

El algoritmo de operación es bastante simple de implementar, ya que solo se basa en algunas operaciones para arrojar un resultado, como datos de entrada espera la lista de posiciones de cada robot y sus orientaciones en el plano de referencia.

Paso 1: Sensado, desde diferentes orientaciones, de las posiciones de los demás robots, dicho proceso se realiza en un numero entero de direcciones y en cada dirección se hace un sensado en las 9 frecuencias de transmisión. Para esto, el robot que está sensando realiza su medición, hace un giro de radianes sobre su propio eje y repite el proceso hasta haber girado 2π radianes.

(1)

Donde:

: Vector de potencial para las i direcciones en las j frecuencias de sensado.

: Vector de intensidades de detección, en las j frecuencias, i direcciones, del robot con el resto de los robots, este varía dependiendo la distancia angular del robot que está sensando respecto al robot lejano, para la obtención de se hace uso del lóbulo de detección del robot que se encuentra sensando.

: Potencial, en las j frecuencias, i direcciones, del robot lejano sobre el robot que está sensando, se calcula haciendo uso de la distancia euclidiana entre ambos robots.

Paso 2: Estimación de

Haciendo uso de se calcula el ángulo de la nueva dirección del robot que está sensando.

\* cos() (2)

\* sen() (3)

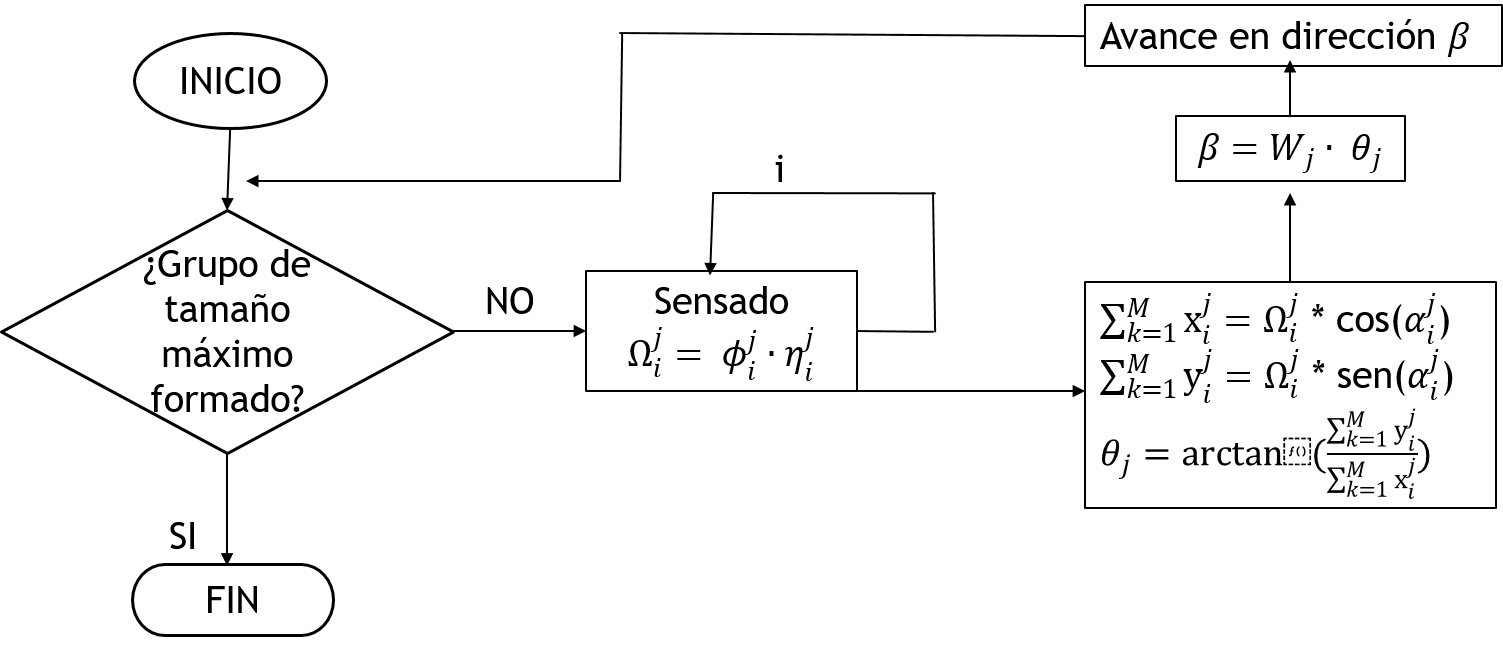
(4)

Teniendo así finalmente un vector con los ángulos de la dirección en donde se encuentra la mayor concentración de robots que emiten en la frecuencia sensada.

Paso 3: Estimación de , el cual es el ángulo de la nueva dirección que adoptara el robot que está sensando:

(5)

Donde es un vector de pesos para las diferentes frecuencias, es decir, es donde se configura el “efecto” que tendrán las concentraciones de robots, que emiten en las diferentes frecuencias, al momento de estimar .



**2 EFECTO DE LAS INCERTEZAS**

Usando el presente simulador, fue posible analizar el comportamiento del problema en un ambiente un tanto ideal, ya que en el software, el giro de los ángulos es correcto (se obtiene un ángulo con las ecuaciones y se gira exactamente a dicho ángulo), el avance es correcto (se avanza la distancia deseada) lo cual asegura un correcto desempeño del algoritmo. Sin embargo, en la realidad esto es casi imposible que ocurra, ya que, debido a la naturaleza del terreno, los sensores y las piezas mecánicas elegidas para armar los robots (tanto de prueba como finales) no permiten movimientos perfectos y precisos.  
  
 Al ser un suelo no rígido (el regolito lunar en específico deja una capa de polvo móvil en su superficie) el moverse, ya sea en forma circular (girar) o recta (avanzar), implica una serie de imperfecciones que se traducen en un error con respecto a lo que internamente calcula el robot que debería moverse, eso aunado a que los sensores que utiliza el robot para percibir al resto de los integrantes del enjambre, son susceptibles a ruido e interferencia por parte del medio, haciendo que inclusive la misma medición esté incorrecta desde el principio.  
  
 Actualmente se está dotando al software con la capacidad de simular estas condiciones, para determinar el nivel de impacto de dichas incertezas en la operación de cada robot y analizar de qué manera influye en el desarrollo de la misión, hasta el momento se tienen en fase de pruebas dos incertezas:

1. Incerteza al girar: Consiste en que, después de determinar el ángulo al que debe girar un robot para posteriormente avanzar, en lugar de usar dicho ángulo, se utilice un número aleatorio que provenga de una distribución con media, el ángulo que obtuvimos, y desviación configurable.
2. Incerteza al avanzar: Consiste en que, al intentar avanzar, en lugar de utilizar la distancia que se avanza en cada paso, se utilice un número aleatorio que provenga de una distribución con media, la distancia predefinida, y desviación configurable.

Estas incertezas están arrojando datos respecto a que tan robusto es el algoritmo, es decir, una medida de cuanta incerteza debe ser acumulada antes de que sea un problema para concretar la misión.

**3 CAPACIDADES DE CLUSTERIX 4.0**

**3 FUNCIONES C**

**4 FUNCIONES MATLAB**

**AsignadorClase**

function robots = AsignadorClase(robots,N)

Argumentos (robots,N)

robots: Arreglo que contiene todas las instancias de robot creadas.

N: Numero de instancias de robots creadas.

Parámetros (robots)

robots: Arreglo que contiene todas las instancias de robot creadas.

Funcionalidad: Asigna la clase a cada robot de acuerdo al tamaño de cluster al que pertenece

**BarridoFrecuencia**

function thetha = BarridoFrecuencia(robot\_referencia,robots,DELTA\_ROTACION,ALFA\_POTENCIAL,N,SIGMA\_ROTACION)

Argumentos(robot\_referencia,robots,DELTA\_ROTACION,ALFA\_POTENCIAL,N,SIGMA\_ROTACION)

robot\_referencia: Instancia del robot que realiza el sensado de otros robots.

DELTA\_ROTACION: Variable global que establece los radianes que se desplaza (angularmente) cada robot durante el sensado de frecuencias.

ALFA\_POTENCIAL: Exponente de amplificación del factor distancia entre el robot de referencia y el robot sensado.

N: Número de instancias de robots creadas.

SIGMA\_ROTACION: Desviación estándar para el cálculo de incertezas aleatorias al girar.

Parámetros (thetha)

thetha: Dirección de la mayor aglomeración de robots emisores en la frecuencia sensada.

Funcionalidad: Realiza la rotación y el cálculo de un ángulo en una frecuencia

**CargaParametros**

function[REPETICIONES,ITERACIONES,L,N,A\_POTENCIAL,ALFA\_POTENCIAL,R,DELTA\_AVANCE,DELTA\_ROTACION,W,SIGMA\_ROTACION,R\_COLISION,ACTIVAR\_GRAFICOS,TAM\_IDENTIFICADOR]= CargaParametros()

Argumentos ()

Parámetros (REPETICIONES,ITERACIONES,L,N,A\_POTENCIAL,ALFA\_POTENCIAL,R,DELTA\_AVANCE,DELTA\_ROTACION,W,SIGMA\_ROTACION,R\_COLISION,ACTIVAR\_GRAFICOS,TAM\_IDENTIFICADOR)

REPETICIONES: Numero de experimentos a realizar.

ITERACIONES: Numero de iteraciones por experimento.

L: Longitud del área de lanzamiento de medidas L x L

N: Número de instancias de robots creadas.

A\_POTENCIAL: Constante de ajuste de amplitud del potencial

ALFA\_POTENCIAL: Exponente de amplificación del factor distancia entre el robot de referencia y el robot sensado.

R: Radio de la circunferencia que circunscribe a cada robot.

DELTA\_AVANCE: Variable global que establece las unidades de distancia que se desplaza cada robot en la dirección de la mayor aglomeración de robots.

DELTA\_ROTACION: Variable global que establece los radianes que se desplaza (angularmente) cada robot durante el sensado de frecuencias.

W: Pesos de las distintas frecuencias

MEDIA\_RUIDO = Media de la función de distribución del ruido de transmisión.

SIGMA\_POTENCIAL Sigma para la función de distribución del ruido de transmisión.

SIGMA\_ROTACION: Desviación estándar para el cálculo de incertezas aleatorias al girar.

R\_COLISION: Distancia necesaria para considerar que dos robots se encuentran en contacto. Se encuentra expresado como una proporción de R.

ACTIVAR\_GRAFICOS: Bandera para indicar si se desean desplegar los gráficos en tiempo real.

TAM\_IDENTIFICADOR: Establece la longitud del identificador de los archivos de salida.

Funcionalidad: Carga de los parámetros iniciales de entrada en forma de variables globales.

**Clustered**

function y=Clustered(x1,x2,R\_COLISION)

Argumentos (x1,x2,R\_COLISION)

x1: Instancia de robot.

X2: Instancia de robot

R\_COLISION: Distancia necesaria para considerar que dos robots se encuentran en contacto. Se encuentra expresado como una proporción de R.

Parámetros (y)

y: Valor booleano que expresa si dos robots se encuentran en un mismo cluster.1 es firmativo,0 es negativo.

Funcionalidad: Determina si dos robots se encuentran en un mismo cluster

**ConteoClusters**

function clusters=ConteoClusters(robots,N)

Argumentos (robots,N)

robots: Arreglo que contiene todas las instancias de robot creadas.

N: Numero de instancias de robots creadas.

Parámetros (clusters)

clusters: Arreglo que contiene la distribución de los distintos clusters formados.

Funcionalidad: Determina cuantos clusters existen y de que tamaño es cada uno.

**CreaArchivo**

function archivo=CreaArchivo(identificador,tipo)

Argumentos (identificador,tipo)

identificador: Establece el identificador del archivo que se va a crear.

tipo: Número que indica el tipo de contenido que albergará el archivo. Los tipos existentes son:

1:Parámetros iniciales  
2:Histograma  
3:Clusters  
4:Tiempo

Parámetros (archivo)

archivo: Instancia del archivo creado.

Funcionalidad: Realiza la creación de una instancia de archivo de salida.

**CreaIdentificador**

Function identificador=CreaIdentificador(longitud)

Argumentos (longitud)

longitud: Numero entero que especifica el tamaño del identificador a generar.

Parámetros (identificador)

identificador: Cadena de caracteres aleatoria.

Funcionalidad: Realiza la creación de un identificador aleatorio para un archivo.

**DestruyeCluster**

function robots = DestruyeCluster(robots,N,robot\_actual)

Argumentos (robots,N)

robots: Arreglo que contiene todas las instancias de robot creadas.

N: Numero de instancias de robots creadas.

robot\_actual: Instancia de un robot.

Parámetros (robots)

robots: Arreglo que contiene todas las instancias de robot creadas.

Funcionalidad: Destruye un cluster.

**Exito**

function exito=Exito(robots,N)

Argumentos (robots,N)

robots: Arreglo que contiene todas las instancias de robot creadas.

N: Numero de instancias de robots creadas.

Parámetros (exito)

exito: Valor booleano que indica si existe el cluster de tamaño 9.

Funcionalidad: Revisa la existencia de un cluster de tamaño 9.

**GeneradorCircunferencia**

function [x,y]=GeneradorCircunferencia(x1,x2)

Argumentos (x1,x2)

x1: Un arreglo de dos coordenadas que representa un punto del plano.

x2: Un arreglo de dos coordenadas que representa un punto del plano.

Parámetros (x,y)

x: Arreglo que contiene las coordenadas x de los puntos de la circunferencia generada.

y: Arreglo que contiene las coordenadas y de los puntos de la circunferencia generada.

Funcionalidad: Genera los puntos de una circunferencia con radio 10 y centro en [(x1(1)+x2(1))/2,(x1(2)+x2(2))/2]

**Inicializacion**

function robots = Inicializacion(N,L,R,A\_POTENCIAL)

Argumentos (N,L,R,A\_POTENCIAL)

N: Número de instancias de robots creadas.

L: Longitud del área de lanzamiento de medidas L x L

R: Radio de la circunferencia que circunscribe a cada robot.

A\_POTENCIAL: Constante de ajuste de amplitud del potencial

Parámetros (robots)

robots: Arreglo que contiene todas las instancias de robot creadas.

Funcionalidad: Carga de los parámetros iniciales de entrada como variables en el espacio de trabajo.

**Inicio**

Función principal en la que inicia la ejecución del simulador.

**Lobulo**

function a=Lobulo(thetha)

Argumentos (thetha)

thetha: Angulo entre el robot de referencia y el robot sensado.

Parámetros (a)

a: Número flotante que indica la intensidad con la que el robot referencia percibe al robot sensado.

Funcionalidad: Calcula la atenuación con la que un robot percibe a otro utilizando el lóbulo uno.

**LobuloCos**

function a=Lobulo(thetha)

Argumentos (thetha)

thetha: Angulo entre el robot de referencia y el robot sensado.

Parámetros (a)

a: Número flotante que indica la intensidad con la que el robot referencia percibe al robot sensado.

Funcionalidad: Calcula la atenuación con la que un robot percibe a otro utilizando el lóbulo dos.

**Potencial**

function pot=Potencial(x1,x2,A\_POTENCIAL,ALFA\_POTENCIAL,MEDIA\_RUIDO,SIGMA\_POTENCIAL)

Argumentos (x1,x2,A\_POTENCIAL,ALFA\_POTENCIAL)

x1: Un arreglo de dos coordenadas que representa un punto del plano.

x2: Un arreglo de dos coordenadas que representa un punto del plano.

A\_POTENCIAL: Constante de ajuste de amplitud del potencial

ALFA\_POTENCIAL: Exponente de amplificación del factor distancia entre el robot de referencia y el robot sensado.

MEDIA\_RUIDO = Media de la función de distribución del ruido de transmisión.

SIGMA\_POTENCIAL Sigma para la función de distribución del ruido de transmisión.

Parámetros (pot)

pot: Número flotante que representa el potencial.

Funcionalidad: Calcula la función de potencial de un vector sobre otro

**RevisaCluster**

function mayor = RevisaCluster(robots,N,robot\_actual)

Argumentos (robots,N)

robots: Arreglo que contiene todas las instancias de robot creadas.

N: Numero de instancias de robots creadas.

robot\_actual: Instancia de un robot.

Parámetros (mayor)

mayor: Valor booleano que determina si el robot actual se encuentra en el mayor cluster formado.

Funcionalidad: Se encarga de revisar si el robot actual pertenece al cluster más grande formado hasta el momento.

**RevisarColision**

function [robots,colision]=RevisarColision(robots,h,l,R\_COLISION)

Argumentos (robots,robots,h,l,R\_COLISION)

robots: Arreglo que contiene todas las instancias de robot creadas.

h: Índice que referencia la posición de un robot dentro del arreglo de robots.

l: Índice que referencia la posición de un robot dentro del arreglo de robots.

R\_COLISION: Distancia necesaria para considerar que dos robots se encuentran en contacto. Se encuentra expresado como una proporción de R.

Parámetros (robots,colision)

robots: Arreglo que contiene todas las instancias de robot creadas.

colision: Variable booleana que indica si hubo colisión.

Funcionalidad: Revisa si existe colisión entre dos robots

**RotacionAvance**

function robots=RotacionAvance   
 (robots,DELTA\_ROTACION,N,W,ALFA\_POTENCIAL,DELTA\_AVANCE,SIGMA\_ROTACION,MEDIA\_RUIDO,SIGMA\_POTENCIAL)

Argumentos (robots,DELTA\_ROTACION,N,W,ALFA\_POTENCIAL,DELTA\_AVANCE,SIGMA\_ROTACION)

robots: Arreglo que contiene todas las instancias de robot creadas.

DELTA\_ROTACION: Variable global que establece los radianes que se desplaza (angularmente) cada robot durante el sensado de frecuencias.

N: Numero de instancias de robots creadas.

W: Pesos de las distintas frecuencias

ALFA\_POTENCIAL: Exponente de amplificación del factor distancia entre el robot de referencia y el robot sensado.

DELTA\_AVANCE: Variable global que establece las unidades de distancia que se desplaza cada robot en la dirección de la mayor aglomeración de robots.

SIGMA\_ROTACION: Desviación estándar para el cálculo de incertezas aleatorias al girar.

MEDIA\_RUIDO = Media de la función de distribución del ruido de transmisión.

SIGMA\_POTENCIAL Sigma para la función de distribución del ruido de transmisión.

Parámetros (robots)

robots: Arreglo que contiene todas las instancias de robot creadas.

Funcionalidad: realiza una iteración del algoritmo para todos los robots

**5 ARQUITECTURA DEL ALGORITMO GENÉTICO**

**Función de adaptación:**

F(w,t)= (w \* e ) – (0.5 t)

e= <-100,100,300,500,800,……..>

w: Vector de pesos

Con esta función de adaptación se considera más apto a un individuo en función de los clusters que pudo lograr usando cierta distribución de pesos.

Se resta adaptación por cada robot que no logró unirse a un cluster y entre mayor sea el tiempo que el sistema tardó en converger.

**Selección:**

Torneo

**Presencia de Mutación : Si**

+/- a

**Cruce:**

**X1 x2 x3 x4 x5 x6 x7 x8 x9**

**y1 y2 y3 y4 y5 y6 y7 y8 y9**

**x5 x6 x7 x8 x9**

**O**

**y5 y6 y7 y8 y9**

**REFERENCIAS**

[1] Kim-Fung Man, Kit-Sang Tang, Sam Kwong. Genetic Algorithms: Concepts and Designs. Hong Kong: Springer 1999.

[2] Gen Mitsuo, Cheng Runwei. Genetic Algorithms and Enginnering optimization. Canada: Wiley & Sons 2000.