UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE INGENIERÍA

Estimación de movimiento de objetos a partir de una secuencia de imágenes.

T E S I S

que presenta para obtener el grado de MAESTRO EN INGENIERÍA

Lic. Juan Carlos Quiroz Sánchez

DIRECTOR DE TESIS:
Dr. Miguel Bravo Zanoguera

MEXICALI, B. C. Diciembre de 2010

RESUMEN de la Tesis de Juan Carlos Quiroz Sánchez, presentada como requisito parcial para la obtención del grado de MAESTRO EN INGENIERÍA. Mexicali, Baja California, México. Junio de 2010.

Estimación de movimiento de objetos a partir de una secuencia de imágenes.

Resumen aprobado por:

_____

Dr. Miguel Bravo Zanoguera
Director de tesis

El seguir el movimiento de un objeto es una tarea esencial en los sistemas de visión artificial, en donde se extrae información a partir de una secuencia de imágenes. En este trabajo de tesis se utiliza un método de segmentación de imágenes para ubicar la posición de los objetos sin la necesidad de conocer la orientación, área o alguna otra característica de los objetos, por lo que se utilizan áreas aproximadas por un grupo de pixeles, caracterizadas por un "centro de masa". Para determinar el movimiento se utiliza la diferencia simple entre centroides de la secuencia y un umbral de desplazamiento máximo. Se emplean diferentes funciones de Matlab que, aunque sirven para el procesamiento de imágenes, no han sido diseñadas para caracterizar el movimiento observado en escenas dinámicas (video), contribuyendo con su propia problemática a la automatización de este proceso. Como objetivo general se busca obtener las trayectorias individuales del movimiento 2D de los objetos que aparecen en una secuencia de imágenes. Se seleccionaron dos secuencias para análisis, en una de ellas las imágenes son en tonos de gris y los objetos (hormigas) tienen diferentes trayectorias, velocidades y desplazamientos, e inclusive un objeto sale del campo visual. La segunda secuencia es de color, representa el movimiento de unos ejes de robots, y algunos objetos entrecruzan su trayectoria. Primeramente se determina un umbral adecuado para delimitar los objetos del fondo, para después detectar y etiquetar los objetos de cada imagen. Para resolver posibles conflictos y determinar la trayectoria individual, se usa un criterio de comportamiento dinámico y el seguimiento cuadro a cuadro de todas las etiquetas generadas inicialmente. Específicamente se logra lo siguiente:

Se obtiene una plataforma de software para el seguimiento de objetos que aparecen en un fotograma consecutivo, que considera la dependencia temporal de las imágenes en el video.

Se reduce la cantidad de información al convertir datos imágenes a datos de seguimiento por coordenadas (matriz de centroides), que representa la trayectoria de objetos.

Se evalúa y notifica la perdida de objetos

Se corrige el intercambio de etiquetas (cruce de objetos) entre cuadros de imagen que ocurre con el uso de software para imágenes estáticas.

A Dios por darme la vida, salud, sabiduría, inteligencia y fortaleza.

A mis padres que formaron parte de mi existencia y mi carácter.

Concepción y Enrique.

A mis hermanos que fueron ejemplo a seguir.

Armando y Martha.

A mi esposa que me apoyó y animó para seguir adelante.

Alicia.

A mi director de tesis por compartirme sus conocimientos y experiencia.

Miguel

ÍNDICE

Índice de figuras

Índice de tablas

Introducción

El desarrollo de métodos eficientes para calcular el movimiento y la estructura de objetos a partir de una secuencia de imágenes es una de las áreas de interés en el campo de la visión artificial. La estimación del movimiento usando imágenes tiene una amplia gama de aplicaciones, que pueden ser clasificadas dentro de tres grandes grupos [1]:

1 Recuperación de la estructura a partir de secuencias de imágenes
2 Compresión y reconstrucción de secuencias de imágenes
3 Seguimiento y caracterización dinámica de objetos en movimiento.

El seguimiento del movimiento es una tarea esencial en cualquier sistema de visión artificial y éste es extraído de la información contenida a partir de una secuencia de imágenes. En la mayoría de los casos, el único dato disponible para realizar este proceso es la variación espacial y temporal del patrón de brillo de la imagen; lo que dificulta el desarrollo de estos sistemas.

Básicamente hay dos formas de determinar el movimiento a partir de secuencias de imágenes: el método de los gradientes y el método de segmentación de imágenes [1]. El método de segmentación de imágenes contiene tres algoritmos básicos: a) segmentación en líneas que extraen el contorno de los objetos, b) segmentación en bloques que modelan las imágenes dividiéndolas en rectángulos y c) segmentación de grupos que modelan las imágenes como áreas amorfas caracterizadas por un “centro de masa” [1]. En éste trabajo se utiliza un algoritmo de segmentación de imágenes del tipo c), ya que solamente se requiere determinar la posición de los objetos a caracterizar por un punto en el espacio, no es necesario conocer la orientación, área u otra característica de los objetos.

Los sensores de imagen se usan para observar el mundo desde el punto de vista de las computadoras que es distinto al nuestro, y el análisis de imágenes puede darnos características y propiedades que nos resultan útiles, según sea la aplicación, como área, orientación, centroide, número de objetos, etc. Con la ayuda de la computadora se puede ampliar el rango de visión, colores y en algunos casos eliminar la subjetivad de lo que se

esta observando que nos distraería de los rasgos importantes que aportan mucha información. La persona que realiza el tratamiento de imágenes es la responsable y tiene la última palabra al dar una interpretación y conclusión de lo que estamos observando.

En los sistemas de inspección industrial de alta tecnología, así como en la investigación moderna de la biología y en los diagnósticos médicos, se tiende a contar con microscopios computarizados y el análisis automático [2]. Muchas tareas rutinarias ya se realizan de manera eficiente con los sistemas automatizados: barrido de la muestra, segmentación y clasificación de objetos. La mayoría de estos sistemas automatizados se basan en la escala de grises o de color de imágenes estáticas. Mientras que el análisis de video se ha vuelto una técnica indispensable para la biología celular experimental, el estudio de animales en tiempo real y para visión robótica en la industria, contribuyendo con su propia problemática a la automatización.

El movimiento aún durante periodos cortos de observación, las inestabilidades mecánicas y la fluctuación de temperatura dificultan la captura y análisis automático de video. Durante la captura de cuadros de video por retardos predefinidos (*time-lapse*) o fotogramas consecutivos, requieren de un ajuste constante del campo de imagen y de la posición de foco para mantener nitidez de los objetos de interés centrados en el volumen de imagen. Esta limitación puede ser superada con el seguimiento de objetos de manera completamente automática utilizando el centro de masa de los objetos. La presente tesis es un ejemplo del empleo de herramientas de Matlab que sirven para el procesamiento de imágenes estáticas, pero utilizadas en la estimación de movimiento de objetos a partir de una secuencia de imágenes y caracterizar el movimiento observado.

Como objetivo general se busca obtener las trayectorias individuales del movimiento 2D de los objetos que aparecen en una secuencia de imágenes I(x,y,t). Para alcanzar el objetivo necesitamos determinar el formato de las imágenes y conocer las operaciones aplicables, así como las diferentes herramientas para su procesamiento junto con la interpretación final. Se seleccionaron dos secuencias de dos tipos de objetos para su análisis, en una secuencia se tienen imágenes en tonos de gris y los objetos son hormigas

que tienen diferentes trayectorias, velocidades y desplazamientos e inclusive un objeto sale del campo visual. La segunda secuencia es de color, representa el movimiento de unos ejes de robots, y algunos objetos cruzan su rastro de trayectoria. Primeramente se determina un umbral adecuado para delimitar los objetos del fondo, para después detectar y etiquetar los objetos y así determinar la trayectoria individual en base a criterios de comportamiento dinámico y el seguimiento cuadro a cuadro de las etiquetas, con lo que se resuelven posibles conflictos de trayectoria.

Como tareas específicas se busca:

1. Obtener de una plataforma software para imágenes estáticas (Matlab), el seguimiento de objetos que aparecen en un fotograma consecutivo, que incluya conocimiento de la dependencia temporal de las imágenes en el video.
2. Reducir la cantidad de información al convertir datos imágenes a datos de seguimiento-coordenadas, que sea precisa a la trayectoria de objetos gráficamente y su respectiva matriz de centroides.
3. Evaluar y notificar la perdida de objetos fuera de cuadro visual
4. Corregir la confusión en el cruce de objetos que ocurre en software para imágenes estáticas.

Este documento, después de su introducción, está compuesto por un grupo de capítulos que se describen a continuación. En el Capítulo 1 “Antecedentes y enfoque teórico”, se describe brevemente otros trabajos sobre este tema y las herramientas de software que se usan, cuales fueron las condiciones de captura y el formato de imagen. El tratamiento de la imagen es con el uso de Matlab y las funciones que tiene integradas para procesamiento de imágenes. Las situaciones posibles que se consideran al aplicar las funciones son desaparición de objetos, fusión de objetos y variación de tamaño. En el Capítulo 2 “Materiales y métodos”, se utilizan dos secuencias de imágenes para el análisis de movimiento a las cuales se les aplican las funciones de MATLAB para el procesado de imágenes contenidas en el *Image Processing Toolbox* (IPT), que es una colección de funciones que extienden las capacidades de MATLAB en el ambiente computacional numérico [3]. Empezando a aislar los objetos en forma individual haciendo la detección de

bordes, eliminando el ruido (información innecesaria) para asignar un punto que los represente (centroide); teniendo una matriz de centroides por imagen, y continuando con trazar las trayectorias individuales de los objetos para finalmente tener una matriz de centroides por objeto que muestra el seguimiento de la trayectoria. Con lo que se reduce la cantidad de información de los objetos de la imagen original, además de corregir el caso en que los objetos están próximos a cruzarse lo que nos proporcionaría falsas trayectorias. En el Capítulo 3 se presentan los resultados de las trayectorias obtenidas del video de hormigas y de los ejes de robot, tanto las tablas de datos como las imágenes de trayectoria. Por último, se presenta las conclusiones, la bibliografía y los apéndices.

Capítulo 1

Antecedentes y Enfoque Teórico

1.1 Antecedentes

A continuación describimos el rumbo de la presente tesis sin abordar de manera detallada aspectos técnicos de cómo se hacen o realizan los diferentes procesos. En trabajos anteriores de investigación sobre detección de movimiento, aunque la problemática técnica inicial pueda ser similar (correspondencia de una imagen a otra, y detección de posición y orientación), la aplicación y los métodos de solución pueden ser muy diversos. Por ejemplo, en algunos sistemas de visión se utilizan algoritmos con un enfoque predictivo para realizar el seguimiento de múltiples robots que juegan fútbol. El objetivo es controlar robots y obtener un rendimiento cercano a la velocidad de vídeo utilizando hardware convencional. El algoritmo utiliza un filtro para eliminar el ruido y estimar recursivamente la posición de los objetos sobre el campo. Se presta especial atención a solucionar los problemas derivados de una iluminación no uniforme del terreno de juego y el seguimiento con marcas de colores. Para lo cual utilizan métodos sencillos con varios canales de señales de video, ya sea el nivel de gris en conjunto con el canal de saturación, o los espacios de color RGB, para realizar la segmentación de la imagen [4].

La estimación del movimiento en una secuencia de imágenes por medio del cálculo del flujo óptico ha sido de amplio interés por algún tiempo, y en la actualidad se busca mejorar el rendimiento de los algoritmos con vistas a su utilización en tiempo real. El flujo óptico se basa en que la información del movimiento de los objetos esta contenida en los cambios de intensidad de la imagen, por lo que se utiliza un método basado en los gradientes espacio-temporales, que determina los cambios espaciales y temporales del patrón de grises de la imagen y a partir de ellos obtiene el flujo óptico (vector por píxel). Tiene un amplio campo de aplicaciones tales como la modelización del entorno, la compresión de video, así como el seguimiento de objetos [5]. Como ejemplo de flujo óptico, en uno de los proyectos investigados, una cámara de video analiza la posición de los dedos de la mano en un instrumento musical, dando seguimiento aun con interferencia

visual debido a la iluminación y sombras. A pesar de no haber obtenido una solución óptima, se va mejorando un seguidor robusto, que pudiera funcionar en tiempo real [6].

Otras aplicaciones más complejas de estimación de movimiento introducen la geometría de proyección para relacionar el movimiento en la imagen con el movimiento del objeto, tal que pueda estimarse la profundidad en el movimiento [7]. O en otros casos también complejos, utilizando una función de probabilidad de flujo óptico se busca estimar los campos de flujo temporal por la dinámica de las texturas, cuyo movimiento difiere radicalmente de los cuerpos rígidos [8]. Pero en general, existen varios trabajos teóricos de aplicaciones y tratamiento de imágenes que presentan los pasos a seguir para la determinación del movimiento a partir de secuencias de imágenes, en los que uno se puede basar para después enfocarse a una aplicación en especial. En estos trabajos de tipo tutorial, básicamente se presentan dos formas de determinar el movimiento a partir de secuencias de imágenes, la determinación del flujo óptico por gradientes y la determinación del movimiento por segmentación de imágenes. Actualmente, una tendencia en las investigaciones es diferenciar los métodos de software o arquitecturas de hardware para estimar el movimiento en tiempo real u optimizar alguna característica [9].

1.2 Procesamiento de imágenes con MATLAB

1.2.1 Filtrado

Para filtrar con Matlab, una imagen se maneja como una matriz de datos con valores de los tonos de gris. Y el aplicar un filtro lineal es la operación de una matriz llamada máscara (kernel o filtro) que multiplica a la matriz imagen, a través de un proceso que se llama convolución. En la convolución, el valor de un pixel transformado se obtiene como suma ponderada de pixeles vecinos. A continuación se muestra como calcular el píxel de coordenados (2, 5), con valor 6, con los pasos siguientes:

- Se rota la máscara de convolución 180 grados alrededor del elemento pixel de interés.
- Multiplica cada par de valores de la matriz imagen y la máscara.
- El nuevo valor de la matriz de imagen es la suma de cada multiplicación.

Matriz de la imagen

Valores de los pixeles de la imagen

12	3	8	20	15	7
2	9	1	14	6	10
16	2	13	5	18	6
21	4	11	9	12	3
1	7	10	8	22	24

4	**1**	**9**
2	**5**	**3**
6	**7**	**8**

Máscara de convolución.

Multiplicación de la matriz imagen por la máscara de convolución rotada.

12	3	8	20 • **8**	15 • **7**	7 • **6**
2	9	1	14 • **3**	6 • **5**	10 • **2**
16	2	13	5 • **9**	18 • **1**	6 • **4**
21	4	11	9	12	3
1	7	10	8	22	24

De aquí el valor del píxel (2, 5) de salida es:

20•8 + 15•7 + 7•6 + 14• 3 + 6•5 + 10•2 5•9 + 18•1 + 6•4 = 486

1.2.2 Centrioide

El centro de área en imágenes binarias es el mismo que el centro de masa si consideramos la intensidad de un punto como la masa de ese punto. Para calcular la posición del objeto, usamos:

$$\boldsymbol{A} = \sum_{i=1}^{n}\sum_{j=1}^{m} B[i,j] \quad \bar{x} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} j\, B[i,j]}{A} \qquad \bar{y} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} i\, B[i,j]}{A}$$

Donde $\bar{x}$ y $\bar{y}$ son las coordenadas del centro de la región medida con respecto al píxel superior izquierdo, A es el área del objeto binario, y B[i,j] es el valor binario del pixel en las coordenadas (i,j).

Estos son los momentos de primer orden. La posición calculada usando los primeros momentos no es necesariamente un entero y usualmente resulta entre los valores enteros de los índices de un arreglo de imagen. Enfatizando que no implica que la posición calculada

es mejor que la resolución de las coordenadas el píxel. Por ejemplo, el centroide de una región binario es el promedio de la localización de píxeles en una región R.

	1	2	3	4	5	6	7	8	10
1									
2		1	1	1	1	1			
3			1	1	1	1			
4				1	1	1			
5					1	1			
6						1			
7									
8									
10									

Para obtener el valor de $\bar{x}$, en la columna x = 2, hay un 1, en la columna x = 3 hay dos 1, en la columna x = 4 hay tres 1, x = 5 son cuatro 1, x = 6 son cinco 1. La suma de cada columna por la cantidad de 1´s entre la cantidad de unos resulta x_c. De igual manera para los valores de $\bar{y}$. Para $\bar{y}$, en la fila y = 2 hay cinco 1, así sucesivamente.

Se hace una suma ponderada de los 1´s de una imagen binaria, y se obtiene el centroide.

$$\bar{x} = \frac{1}{15}\left[2 + (2 \cdot 3) + (3 \cdot 4) + (4 \cdot 5) + (5 \cdot 6)\right] = 4.67$$

$$\bar{y} = \frac{1}{15}\left[(5 \cdot 2) + (4 \cdot 3) + (3 \cdot 4) + (2 \cdot 5) + 6\right] = 3.34$$

1.3 Tipo y análisis de imágenes.

Después de esta exposición de antecedentes, creemos conveniente describir el entorno que distingue nuestro trabajo de otros ambientes de visión por computadora. Encontramos en nuestras secuencias de imagen que no tenemos graves problemas de iluminación, se puede considerar que el campo visual ha sido iluminado uniformemente. Nuestro trabajo no es en tiempo real, sino mas bien trabajamos fuera de línea después de una grabación de objetos en un ambiente controlado; sólo nos interesa describir las trayectorias de puntos de interés. No usamos color, reducimos a nivel de gris en el caso de

video en color, o utilizamos video monocromático. No necesitamos geometría de proyección sino que obtenemos resultados en coordenadas del espacio 2D de imagen, y podemos hacer un ajuste de escala simple. Lo que buscamos es tener un sistema de metrología por imágenes, y no se intenta obtener un sistema de control, por ejemplo: no controlamos robots sino que medimos su desempeño. Básicamente, nuestros problemas son cruces simples de objetos sin llegar a una fusión de ellos, pero esto altera la identificación de objetos por el algoritmo de búsqueda de objetos y tenemos que corregir esos errores.

Cuando tenemos una función entre la intensidad y las coordenadas espaciales, podemos decir que tenemos una imagen. Esta imagen puede estar compuesta de tonos de gris o de color y la forma de poder manipularla con un computador es en forma matricial bidimensional o tridimensional. Y como tal se le pueden aplicar todas y cada una de las operaciones matemáticas. Una imagen puede ser definida como una función bidimensional f(x,y), donde x & y con coordenadas en el plano, y la amplitud de f para cualquier par de coordenadas (x,y) es llamada intensidad de la imagen de ese punto [3]. Para el seguimiento de objetos, primeramente se filtran las imágenes para solo trabajar con la información necesaria (filtrado, umbralizado), se etiquetan o identifican los objetos (delimitado de regiones, etiquetando regiones), se hace la detección de su posición respecto a una referencia, posición relativa, y se da un seguimiento individual y en algunos casos predicción del movimiento.

El tipo de imagen para el análisis del movimiento se muestra en la figura 1.1. El tipo de formato de la imágenes usadas es *.tif (formato de imagen etiquetada) que es una serie de bloques que conforman la imagen. Estos bloques pueden contener cierta información sobre la imagen en sí, su tamaño, su manejo del color, información a las aplicaciones que utilicen este archivo y texto [9]. Las características de las imágenes de la secuencia son las que se obtienen con la función de MATLAB, *info = imfinfo('Anta08RD.tif')*, como se ve en la Tabla 1 [10]. Es importante tomar en cuenta cada una de estas características, ya que algunas de ellas o varias nos ayudaran a localizar un objeto o varios, etiquetarlo y darle seguimiento en una secuencia de imágenes (movimiento).

Figura 1.1. Tipo de imagen para el análisis de movimiento.

Las imágenes se representan como matrices [3], y Matlab maneja este tipo de imágenes como imágenes de intensidad de tipo monocromáticas que son una matriz de datos de 640 x 480 pixeles, que han sido escalados representando la intensidad en un rango [0, 255] de 256 tonos de gris [3]. Se usan dos tipos de secuencias para el estudio del movimiento. Estas secuencias representan un ejemplo de escenas con movimiento típicas de un ambiente de iluminación controlado en un laboratorio, aunque no han sido optimizados los parámetros fotométricos, se importan las imágenes al contexto Matlab [10].

Para esta tesis la primera secuencia de imágenes es un grupo de hormigas moviéndose en un área, cada una de las imágenes está en tonos de gris. Para etiquetar cada una de las hormigas hay que determinar primero un punto que las represente y antes que todo hay que eliminar de las imágenes la información o datos que no se ocupan, es decir filtrar espacialmente, además de analizar los diferentes casos de movimiento de los objetos. La segunda secuencia de imágenes muestra los ejes de un brazo robot que está tomando una pieza y la está colocando en otro lugar. Las coordenadas de los ejes muestran la función de posición programada y puede servir para supervisar la operación.

Tabla 1.1. Lista de campos/parámetros de las imágenes y su definición obtenidas con el comando de MATLAB (*imfinfo*)

Campo	**Concepto/Definición**
Filename: 'Anta08RD.tif'	Cadena que contiene el nombre del fichero/imagen
FileModDate:'19-Mar-2004 19:48:52'	Cadena que contiene la fecha de modificación del fichero.
FileSize: 315152	Entero que contiene el número de bytes del fichero.
Format: 'tif'	Cadena que contiene el formato del fichero
FormatVersion:	Cadena o número que describe la versión del formato.
Width: 640	Número de columnas de la imagen.
Height: 480	Número de filas de la imagen.
BitDepth: 8	Entero que indica el número de bits por píxel.
ColorType: 'grayscale'	Imagen en escala de grises
FormatSignature: [73 73 42 0]	
ByteOrder: 'little-endian'	
NewSubfileType: 0	
BitsPerSample: 8	
Compression: 'Uncompressed'	
PhotometricInterpretation: 'BlackIsZero'	
StripOffsets: 7906	
SamplesPerPixel: 1	
RowsPerStrip: 480	
StripByteCounts: 307200	
XResolution: 72	
YResolution: 72	
ResolutionUnit: 'Inch'	
Colormap:	
PlanarConfiguration: 'Chunky'	
TileWidth:	
TileLength:	
TileOffsets:	
TileByteCounts:	
Orientation: 1	
FillOrder: 1	
GrayResponseUnit: 0.0100	
MaxSampleValue: 255	Valor máximo de niveles de gris
MinSampleValue: 0	Valor mínimo de niveles de gris
Thresholding: 1	
NewSubFileType: 0	
DateTime: '2003:03:31 11:49:43'	Día de creación

El uso de Matlab y sus funciones como herramienta para el procesamiento de imágenes en la presente tesis permite diseñar y conjuntar un programa dinámico a las necesidades, de obtener información extrayendo las características de imágenes y obtener resultados sencillos a partir de una secuencia de imágenes. Las reglas generales de búsqueda de información empiezan con delimitar los objetos (por ejemplo: hormigas) en un área individual en imágenes binarias con filtrado y umbralización de la imagen original. Con los objetos de las imágenes delimitados en un área, una función determina un punto característico (centroide) que muestra la posición de cada objeto en cada una de las imágenes de la secuencias. Nos apoyamos en la función *REGIONPROPS* de Matlab que proporciona información como el centroide y área, entre otras características. Para determinar la trayectoria de cada objeto se determina evaluando la cercanía de un punto (centroide) de referencia al más próximo de la siguiente imagen. Con los centroides agrupados por objetos que siguen una trayectoria se generan tablas que muestran la información en forma ordenada y sencilla de la secuencia de imágenes.

Existen situaciones especiales de los objetos en las escenas, que dificultan el uso de MATLAB para análisis de movimiento sobre una secuencia de imágenes. Puede suceder entre cuadros de imagen consecutivos alguna de las siguientes condiciones:

- Que el objeto no se mueva o que todos los objetos se mueven.
- En cada imagen los objetos se mueven en distintas direcciones a diferentes velocidades.
- Que un objeto desaparezca. De los objetos en la primera imagen, un objeto se va desplazando hasta desaparecer, teniendo al final menos objetos.
- Que se fusionen objetos. Algunos objetos se llegan a tocar, y aunque el programa determine una sola área esta se mantiene, es decir el área total es la suma de las individuales.
- Un objeto toca el borde y desaparece.
- Que se deforme un objeto. La región que delimita el área de un objeto se mantiene constante y esta tiene variaciones en su geometría.
- Que un objeto crezca. La región que delimita el área de un objeto tiene un incremento de alrededor del ±10%.

Capítulo 2

Materiales y Métodos

2.1 Secuencias de imágenes bajo prueba

Se usaron dos tipos de secuencias para el estudio del movimiento. Estas secuencias representan un ejemplo de escenas con movimiento típicas con ambiente de iluminación controlado en un laboratorio, aunque no han sido optimizados los parámetros fotométricos.

Secuencia 1 Hormigas. Representa a un grupo de hormigas moviéndose en un área. Son 11 imágenes en tonos de gris (blanco y negro) que contienen 8 hormigas en las 3 primeras imágenes, y 7 hormigas en las 8 imágenes siguientes. El tamaño de imagen es 640 x 480 pixeles, y la secuencia de imágenes fue tomada de un archivo presentado en Internet.

Secuencia 2 Robots. Son 85 imágenes de color que representan el movimiento de un brazo de robot, de tomar una pieza y colocarla en otro lugar. Se observan cuatro ejes del robot, y un eje sirve como punto fijo de referencia para medir el error de detección inicial. De especial interés es la supervisión automática de un brazo robótico GE-Fanuc modelo LR MATE 200i que conforma a este sistema flexible de manufactura (FMS, por sus siglas en inglés). Se utilizó una cámara digital Pentax Optio 450 para la captura de vídeo en formato QuickTime, con tamaño de cuadro de 320 x 240 píxeles y a una velocidad de 15 tramas por segundo.

2.2 Preprocesamiento de secuencia de imágenes de hormigas

2.2.1 Filtrado espacial de imagen

Cuando una imagen es adquirida por una cámara, en ocasiones no es posible usarla directamente, la imagen puede estar distorsionada por variaciones en intensidad, iluminación, o contraste [11]. Es necesario filtrar las imágenes, eliminar la información que no ocupamos para la detección de bordes. Como esto puede ser modelado como un sistema lineal [12], en el cual se tiene una imagen de salida que es la convolución de la imagen de entrada con otra matriz mas pequeña, en nuestro caso se tienen matrices de convolución

(mascara, kernel u operador de convolución) que se aplican sobre cada punto (x,y) de las imágenes [11]. Además, es necesario lograr contraste entre el objeto y el fondo, esto puede requerir de una separación entre los niveles de gris, también llamado ecualización o modificación del histograma de niveles de gris [9]. En la figura 2.1 se muestra una sección de una imagen de la cual podemos ver con más detalle las características del procesamiento en el dominio espacial, que operan directamente en los pixeles de la imagen cuando se define una región de la matriz imagen (vecindad espacial) alrededor de el punto (x, y) [3]. Para accesar y mostrar las imágenes, la función utilizada de Matlab es *imread* con *imshow* [3], en el formato *.tif que reconoce Matlab [3]. Necesitamos eliminar los puntos brillantes de iluminación en el cuerpo de las hormigas y eliminar las patas de las hormigas, para integrar en un solo objeto el cuerpo, además de eliminar las sombras que podrían mostrar un doble objeto. El filtrado debe conseguir diferenciar entre un objeto y el fondo, para que el proceso de umbralización nos genere fácilmente una imagen binaria que corresponda con los objetos que existen en la escena.

Para seleccionar el filtro, nos dimos a la tarea de probar funciones de filtrado que proporciona Matlab y variar los parámetros de la función; analizando el resultado obtenido ampliando una sección característica de la problemática observada y usando la teoría de filtros de orden [12]. La selección es enteramente cualitativa y las diferencias o resultados las obtenemos a partir de las imágenes de salida al aplicar el filtro. Existen otros filtros, pero son para situaciones de manejo de color, contornos, ruidos “sal” y “pimienta”. Las características de este proceso de búsqueda de mejor filtro se presentan a continuación:

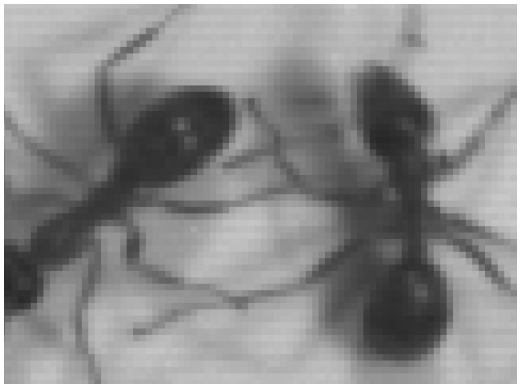
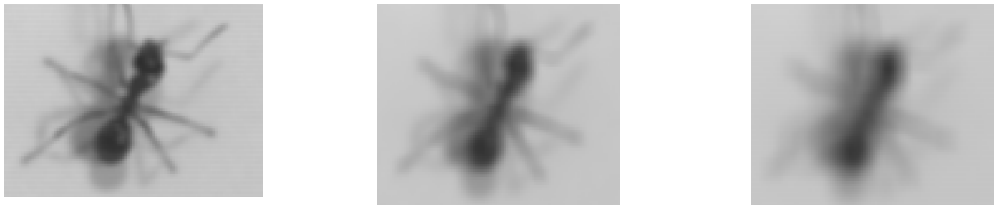


Figura 2.1. Sección de imagen ampliada.



a) b) c)

Figura 2.2. Resultado de aplicar el filtro *imfilter(I,h)*, com parámetro a) n = 3, b) n=6 y c) n =9 .

A) El primer filtro que se utiliza es *imfilter(**I**,h)*; donde ***I*** es la matriz imagen y $h = ones(n,n)/n^2$ es la matriz cuadrada de valores unitario por la cual se multiplica cada píxel de la imagen. El valor de salida del píxel es una combinación lineal de los valores de los píxeles vecinos del píxel de entrada. Tal filtro es llamado filtro promediador, obteniendo el resultado en una sección ampliada que se muestra en la figura 2.2, y en resumen: con n = 3 no elimina los puntos blancos de la iluminación en el cuerpo de las hormigas, ni las patas; con n = 6 disminuyen más los puntos de la iluminación, pero se hace muy delgado el cuerpo y las patas aun presentes; y con n = 9 prácticamente se eliminan los puntos de la iluminación, pero obtenemos de un cuerpo dos partes, además se vuelve muy borroso el objeto.

B) Se utiliza ahora la mascara *h = fspecial('unsharp'),* obteniendo la imagen de la figura 2.3. Con este filtro se tiene una buena definición de los detalles de los objetos, se realzan o remarcan las patas, el cuerpo, las sombras, los puntos blancos de iluminación en el cuerpo de las hormigas. Pero lo que se requiere es lo contrario, deseamos solo obtener

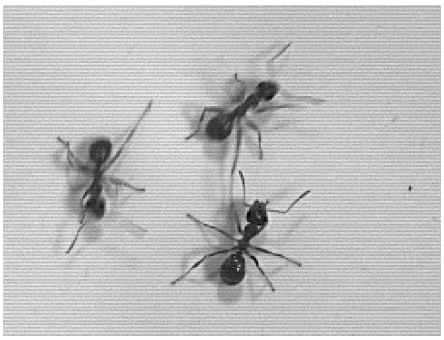


Figura 2.3. Resultado de aplicar *h = fspecial('unsharp').*

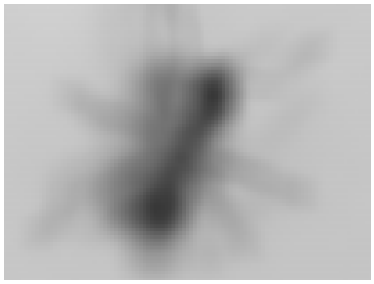


Figura 2.4. Resultado de aplicar *h = fspecial('average', n).*

una región o área individual para cada hormiga tal que podamos identificar con punto característico (centroide). Si se aplica una función de detección de bordes no habrá un área o contorno cerrado para determinar el centroide.

C) Con la función *h = fspecial('average', n),* un filtro de media y *n* = 7 tenemos el mismo efecto que el filtro considerado en el inciso A) anterior, ver figura 2.4.

D) Aplicando *h = fspecial('laplacian', n),* resulta que no hay un contraste entre el fondo y el objeto, además que no elimina las patas, pero si las sombras (ver figura 2.5).

E) Con el filtro h = fspecial('log', 5, 0.3); este filtro no elimina los puntos blancos de iluminación, las sombras o las patas aunque delimita el contorno del objeto y del fondo, pero aun así no seria posible asignar un solo contorno a un objeto (ver figura 2.6).

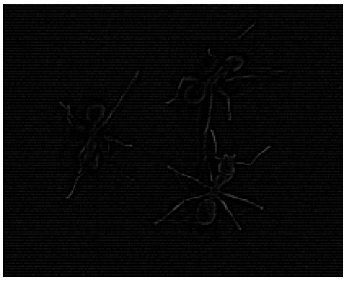


Figura 2.5. Resultado de aplicar *h = fspecial('laplacian', n).*

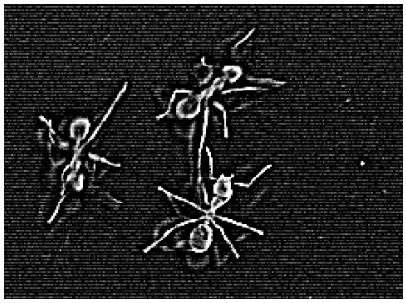


Figura 2.6. Resultado de aplicar *h = fspecial('log', 5, 0.3)* .

F) Usando *f = medfilt2(I, [n n], 'symmetric')* con *n* =7 se muestra en la figura 2.7. Cuando se usa la opción *'symmetric'*, el tamaño de la imagen se extiende en los bordes como si se reflejara en espejo [3]. Este filtro parece ideal, ya que elimina los puntos de alta intensidad y se mantiene el cuerpo de la hormiga en un solo elemento/conjunto. Además que "borra" las patas y se integran a la sombra en un nivel de gris que contrasta diferente al del cuerpo, con lo cual al aplicar un umbral acorde se obtendrán, dentro de una sola área solo el cuerpo de la hormiga, o píxeles interconectados que representan cuerpo de la hormiga, es decir un contorno cerrado (lleno) único a cada objeto, programa *filtromedhormsymetric2.m* o programa *filtromedhorm.m.*

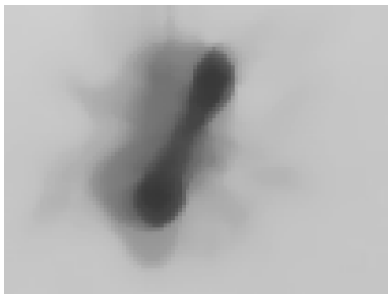


Figura 2.7. Resultado de aplicar *f = medfilt2(I, [n n], 'symmetric')* .

2.2.2 Umbralizado y conversión binaria

Para hacer una separación binaria de los niveles de gris (0 a 255) a un arreglo/matriz de 0´s y 1´s, se realiza un umbralizado [11]. Partiendo de la imagen filtrada, para separar entre el cuerpo de la hormiga y el fondo, se realiza una función umbral (H) con un valor tal, que por debajo de éste los objetos no desaparezcan, se corten o se hagan pequeños, y que por arriba de este valor no se unan dos objetos. Es necesario que los objetos y el fondo tengan un contraste suficientemente amplio [11]; este proceso es la binarización mediante la detección de umbral y extracción de regiones [12].

La imagen es una matriz en tonos de gris con valores desde 0 a 255, y para establecer el umbral correcto, ni demasiado bajo o alto, nos llevo a analizar todas las imágenes filtradas de la secuencia con diferentes umbrales (H de 50 a 150). Llegando a considerar un umbral de H = 100 para separar cada imagen en dos partes, como se muestra en la figura 2.8, donde los valores de intensidad del píxel menores a 100 son negros y mayores a este son blancos. Con el programa *filtromedumbral.m* se investigó el valor de umbral más adecuado. Para valores de H menores a 70, algunos objetos se dividen en varias regiones (no conforman una región única). Mientras que para H mayor a 110 empiezan a aparecer regiones que no pertenecen al objeto, y además se empiezan a fusionar objetos en uno solo.

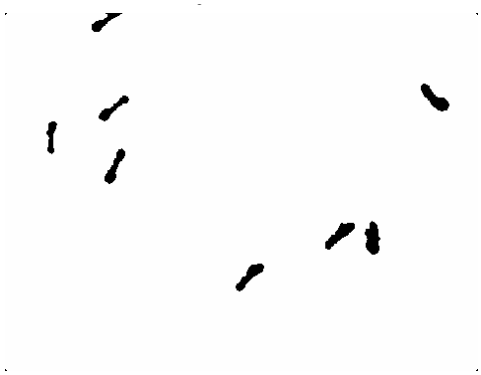


Figura 2.8. Imagen umbralizada con un valor de H = 100.

Se puede considerar que el proceso de filtrado y umbral es correcto cuando se tiene una detección de bordes donde se obtienen áreas que corresponden a cada objeto con una buena ubicación y tamaño. Y con esto podemos determinar los centroides que caractericen a los objetos.

De la imagen umbralizada se pueden detectar regiones en las esquinas que se podrían considerar como objetos según el valor de umbral dado (figura 2.8), pero en realidad no corresponden a objetos reales, sino que son producidos por características propias de la toma de la imagen y la iluminación. Por lo cual se aplica otro criterio sobre la imagen umbralizada, tal que si el objeto es menor a cierta área o número de píxeles entonces forma parte del fondo, es decir les da un valor lógico que corresponde al fondo (1 corresponde al blanco). En este caso si el límite para considerarse un objeto es de 50 píxeles, esto se realiza con el comando *BWAREAOPEN* que remueve objetos pequeños usando morfología matemática [12], programa *closehormcentro.m.*

2.2.3 Detección de centroides

Un centroide es el par de coordenadas (i,j) que corresponden al centro de un objeto, y nos puede servir para reducir la cantidad de datos que serían necesarios para describir un objeto en la imagen, si es que existe un solo tipo de objeto en la escena y sabemos su forma. Para obtener los centroides se utiliza la función de MATLAB: *REGIONPROPS*. La función *REGIONPROPS*, utilizada como *STATS = REGIONPROPS (L,PROPERTIES)*, una matriz que mide propiedades de los objetos que se encuentran en una imagen binaria. En especifico, mide un conjunto de propiedades de cada región (objeto) etiquetada en la matriz L de etiquetas. Elementos enteros positivos de L corresponden a diferentes regiones. *STATS* es una estructura de arreglo matricial determinada por el valor máximo de la matriz L. Los campos del arreglo matricial denotan diferentes propiedades para cada región, y se enlistan a continuación.

Tabla 2.1. Parámetros de la función *REGIONPROPS*.

'Area'	'ConvexHull'	'EulerNumber'
'Centroid'	'ConvexImage'	'Extrema'
'BoundingBox'	'ConvexArea'	'EquivDiameter'
'SubarrayIdx'	'Image'	'Solidity'
'MajorAxisLength'	'PixelList'	'Extent'
'MinorAxisLength'	'PixelIdxList'	'FilledImage'
'Orientation'	'FilledArea'	'Eccentricity'

Para poder usar la función *REGIONPROPS*, primeramente hay que etiquetar la imagen binaria, es decir asignar un número a cada objeto. Esto es posible con la función *BWLABEL*, que en una forma de barrido por los pixeles de la imagen, etiqueta las regiones de pixeles que se encuentran unidos [12]. La función *L = BWLABEL(BW,N)* regresa una matriz L, del mismo tamaño de BW, donde BW es una matriz de una imagen binaria. La matriz L contiene las etiquetas para los componentes conectados en BW. N pude tener un valor de 4 o 8, donde especifica la forma en que los pixeles de un objeto deben estar conectados. La forma como asigna la etiqueta (número de objeto) es haciendo un recorrido de arriba abajo (filas) y de izquierda a derecha (columnas).

Aplicando la función *REGIONPROPS* a la imagen de los objetos etiquetados, obtenemos los centroides de cada región como uno de los campos de información que se calcula para cada región. La siguiente figura 2.9 muestra la posición de centroide para las

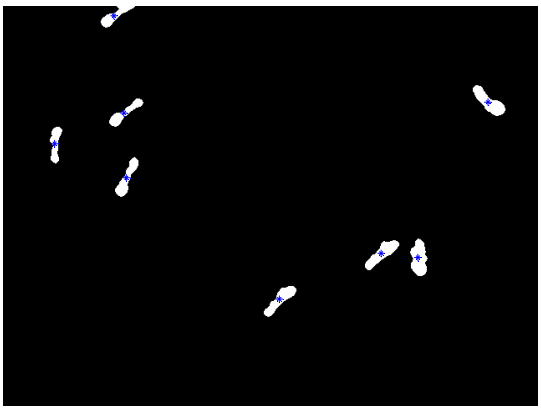


Figura 2.9. Resultado de aplicar la función REGIONPROPS a la imagen umbralizada.

regiones encontradas en la primera imagen, programa *closehormcentro.m.*

Este proceso de filtrado, umbral, remoción de objetos pequeños y obtención de centroides se integra como una función, que tiene como entrada una imagen a procesar I y de salida una matriz "centroids". El tamaño de la matriz de salida será de un número de líneas igual al número de objetos (hormigas), y cada línea con dos elementos (programa *funcentro2.m*).

Si a la imagen original le hubiéramos aplicado directamente la función *REGIONPROPS* para detectar centroides, tendríamos muchas regiones que se considerarían como objetos. La figura. 2.10 muestra este caso, donde se observa una gran cantidad de centroides, por eso se hizo pre-procesamiento.

Cuando el programa diseñado para esta tesis *funcentro2.m* se aplica a cada una de las imágenes obtenemos los centroides (objetos) de cada imagen, y almacenados en una matriz Cn. Este proceso de repetido sobre todas las imágenes es llevado a cabo por el programa *centroconfuncion4.m*, mostrándolo en una sola gráfica, figura 2.11, donde se utiliza un color para centroides pertenecientes al mismo cuadro de imagen.

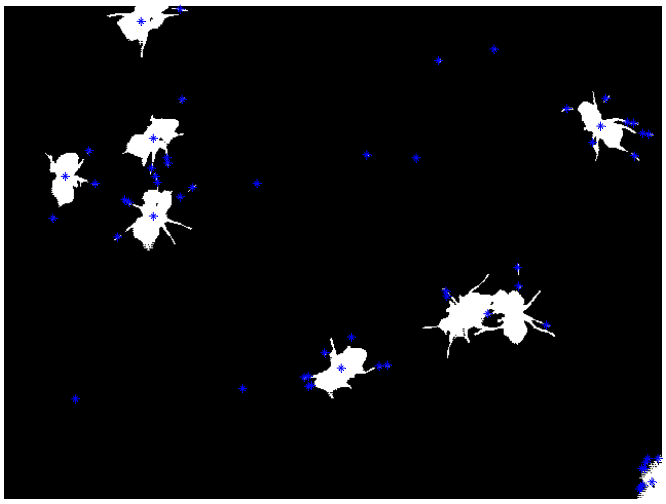


Figura. 2.10. Resultado de aplicar directamente la función REGIONPROPS a la imagen original.

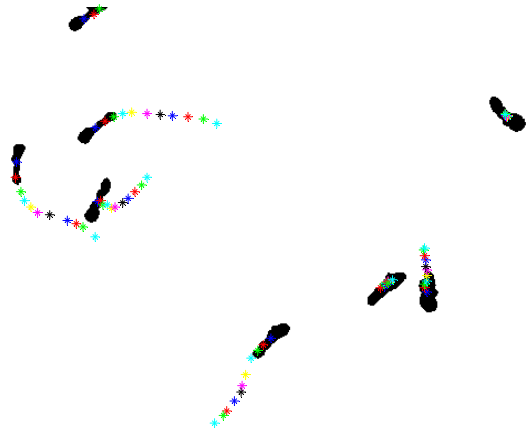


Figura 2.11. Centroides de cada uno de los objetos de cada imagen, generados por la función *centroconfuncion4.m.*

Además se genera una matriz (CT) que contiene todos los centroides de todas las imágenes, pero sin que necesariamente se hubiera desarrollado una relación entre los centroides de cada imagen en la secuencia.

2.3 Seguimiento de trayectoria y análisis de movimiento en la secuencia de imágenes de hormigas

2.3.1 Acumulación de centroides para construir la trayectoria

La imagen de la figura 2.11 muestra la acumulación de centroides encontrados en cada cuadro del video, generados por la función *centroconfuncion4.m*, podemos advertir la trayectoria de cada hormiga, unas recorren mucho mas distancia que otras, no se cruzan o se unen, pero una de ellas sólo aparece en unos cuantos cuadros y después desaparece. Todo lo anterior que nosotros podemos ver claramente, para la computadora solo son puntos en el plano; estos centroides son puntos independientes sin asociación alguna a objeto, ni ligado a otra escena en el tiempo. Si le diéramos a la computadora un centroide de cualquier cuadro, no tiene una guía rápida para escoger las coordenadas que le corresponden en un cuadro consecutivo, sino que debe manipular la masiva cantidad de datos que representa cada imagen.

Ahora, de este grupo de puntos (centroides) ¿cómo los asociamos en trayectorias individuales para determinar distancia y velocidad? Extraemos una submatriz

correspondiente a los centroides de la primera imagen que se usarán como referencias para cada serie de grupos de conjuntos por objeto. Luego tomamos los centroides de la primera imagen y los de la segunda, se miden las diferencias en *x* y *y*, y se determina el valor absoluto entre centroides y grafican las distancias del primer objeto de la imagen a todos los centroides de la segunda imagen; ver figura 2.12. De todas estas mediciones referenciadas al objeto de la primera imagen, se toma la distancia mínima para asociar el centroide que le corresponde en la imagen dos, y así sucesivamente para cada una de las imágenes posteriores y sus objetos de la imagen previa **(**programa *dishorm1.m*).

2.3.2 Análisis de movimiento y matriz de de centroides

Como se va determinando el centroide más cercano de la imagen siguiente a la referencia de la imagen original, se almacenan estos centroides en una matriz dinámica que crece conforme se van encontrando los centroides. Esta relación de secuencia de centroides van trazando la trayectoria de los objetos. Aplicando el proceso anterior de determinar la distancia mínima entre centroides de imágenes sucesivas, obtendríamos las trayectorias individuales de cada objeto (programa *trayecdishorm2.m*). En la figura 2.13 se muestra el resultado de distancias mínimas entre los centroides asociados del primer objeto únicamente.

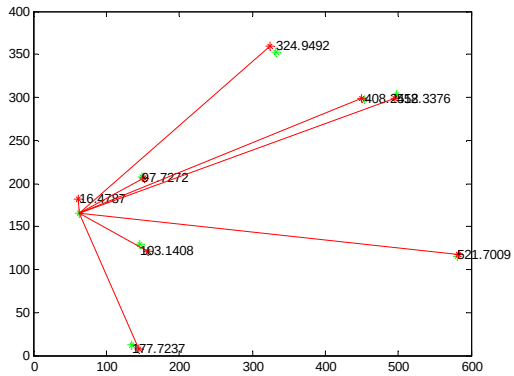


Figura. 2.12. Proceso de medición de distancias a un objeto de referencia. Los centroides de la primera imagen son de color verde.

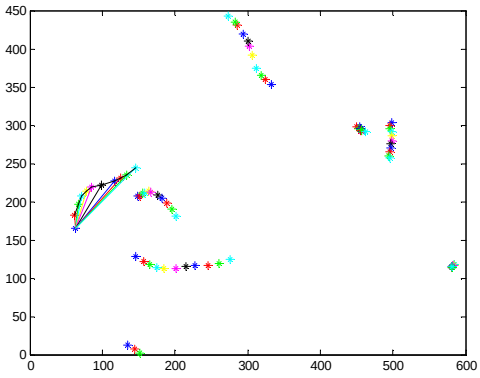
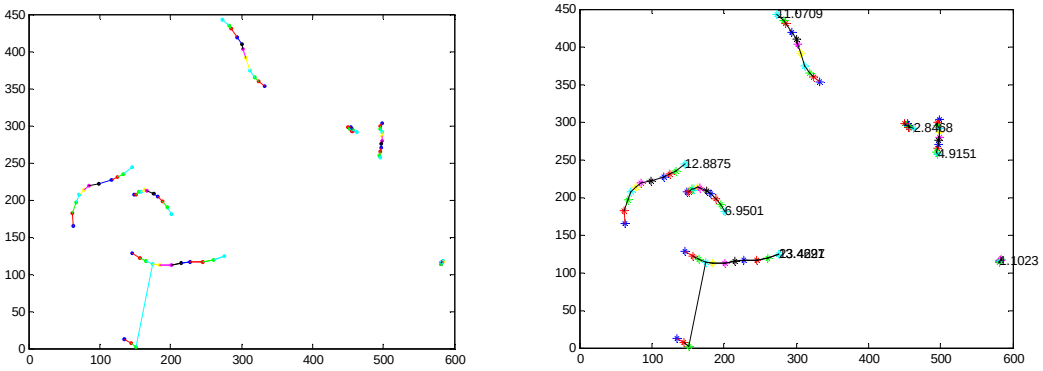


Figura 2.13. Grupo de centroides agrupados para el primer objeto en base a las distancias mínimas.

Aplicado este proceso para determinar así la trayectoria de cada objeto, asociando sus centroides (ver figura 2.14), se puede observar una asociación errónea de una trayectoria de los mismos centroides para dos objetos diferentes.

El programa *trayectoriahorm4.m*, genera la gráfica de trayectorias por objeto, además de en un listado la agrupación de los centroides por objeto a través del uso de una matriz dinámica. Al final se obtiene una matriz que agrupa los centroides por objeto, matriz PT (ver Apéndice I). Podemos observar que la trayectoria del segundo objeto, que sale del campo visual, se confunde o entrelaza con la trayectoria del tercer objeto (figura 2.15). Este



a) b)

Figura 2.14. a) Trayectoria primaria de los objetos determinada por el programa *trayectoriahorm2.m* b) el programa *trayevvalorhorm.m* muestra las trayectorias y el desplazamiento promedio de cada objeto.

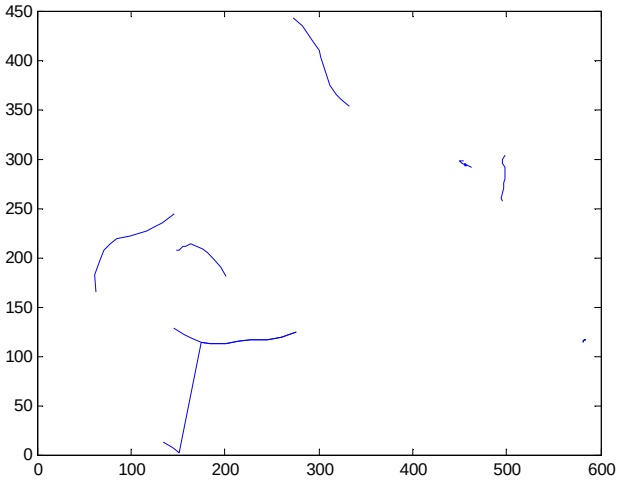


Figura 2.15. Grafica de trayectorias por objeto donde dos objetos tiene asociada una misma trayectoria, programa *trayectoriahorm4.m.*

proceso no es capaz de determinar que un objeto salió del campo visual y termino su trayectoria temprano, sino que le busca una trayectoria, la más corta que se pueda formar de entre la lista de centroides.

2.3.3 Corrección de la trayectoria de objetos fuera de campo visual

Para corregir el error de asociar una misma trayectoria a dos objetos diferentes, hacemos un análisis de desplazamiento de cada objeto en cada secuencia de imágenes. Tenemos los centroides agrupados por objeto, en donde para cada objeto se tomó el centroide que diera el menor incremento entre imágenes, a partir del centroide que ya se conocía. Pero cuando uno de los objetos sale del campo visual, provoca que su trayectoria respectiva se asocie a otra trayectoria de otro objeto. Para solucionar esta situación se establece un radio o distancia máxima para asociar un centroide próximo al objeto de referencia. La distancia máxima se determina analizando el valor máximo de cada objeto y de todos se considera el máximo eliminando el incremento de error de trayectoria. El objeto 1 tiene el incremento máximo de 19.2902, si eliminamos el valor erróneo de 113.5074 del objeto 2. Por lo anterior se toma un valor de 20 píxeles como radio máximo para determinar si un centroide corresponde al conjunto de posiciones de un objeto en su desplazamiento en la secuencia de imágenes. Por lo que en el proceso de determinar la distancia minima entre

el objeto de referencia y el de la siguiente imagen, si se obtiene una distancia entre ellos mayor a 20, se detiene la secuencia de asociación de trayectoria con la función *break* de Matlab [3]. Este proceso de verificación lo incluimos en un nuevo programa (*treyectoriahorm5.m*) obteniendo una gráfica de trayectorias corregida. En la Tabla se presenta una descripción de los programas realizados para la determinación de la trayectoria de los objetos.

Tabla 2.2. Lista de programas desarrollados para la determinación de trayectoria.

	nombre programa (código MATLAB) *.m	**Función, Objetivo, Propósito**	**Entrada**	**Salida**
A	**filtromedhorm.m**	Filtrar imagen **I**	Imagen **I**	Imagen filtrada
B	**filtromedumbral.m**	Obtener la primer imagen umbralizada	Imagen **I**	Imagen umbralizada
C	**closehormcentro.m**	Obtener los centroides de la primer imagen	Imagen **I**	Muestra centroides de primer imagen
D	**funcentro2.m**	Función que genera los centrioides a partir de una imagen de entrada	Imagen **I**	Matriz de centroides de los objetos en la imagen
E	**centroconfuncion4.m**	Obtiene la distancia de un centroide de referencia a los centroides de la siguiente imagen	11 imágenes	Grafica de todos los centroides y matriz de todos los centroides.
F	**dishorm1.m**	Este programa mide y grafica las distancias del 1er objeto la todos de las siguientes imágenes	Submatriz de centroides de la primera imagen y submatriz de centroides de la segunda imagen.	Distancia minima y grafica de medición
G	**trayecdishorm2.m**	Determina trayectoria de los objetos	Matriz CT que contiene todos los centroides de todas las imágenes	Grafica de trayectoria por objeto
H	**trayectoriahorm5.m**	Grafica la trayectoria de los objetos a través de mediciones sucesivas entre los centroides de cada imagen, determinando distancias mínimas de centroides, teniendo como entrada matriz de centoides por imagen y salida matriz de centroides por objeto. Corrigiendo trayectorias falsas al desaparecer un objeto.	Matriz CT de centroides por imagen	Matriz PT de centroides por objeto

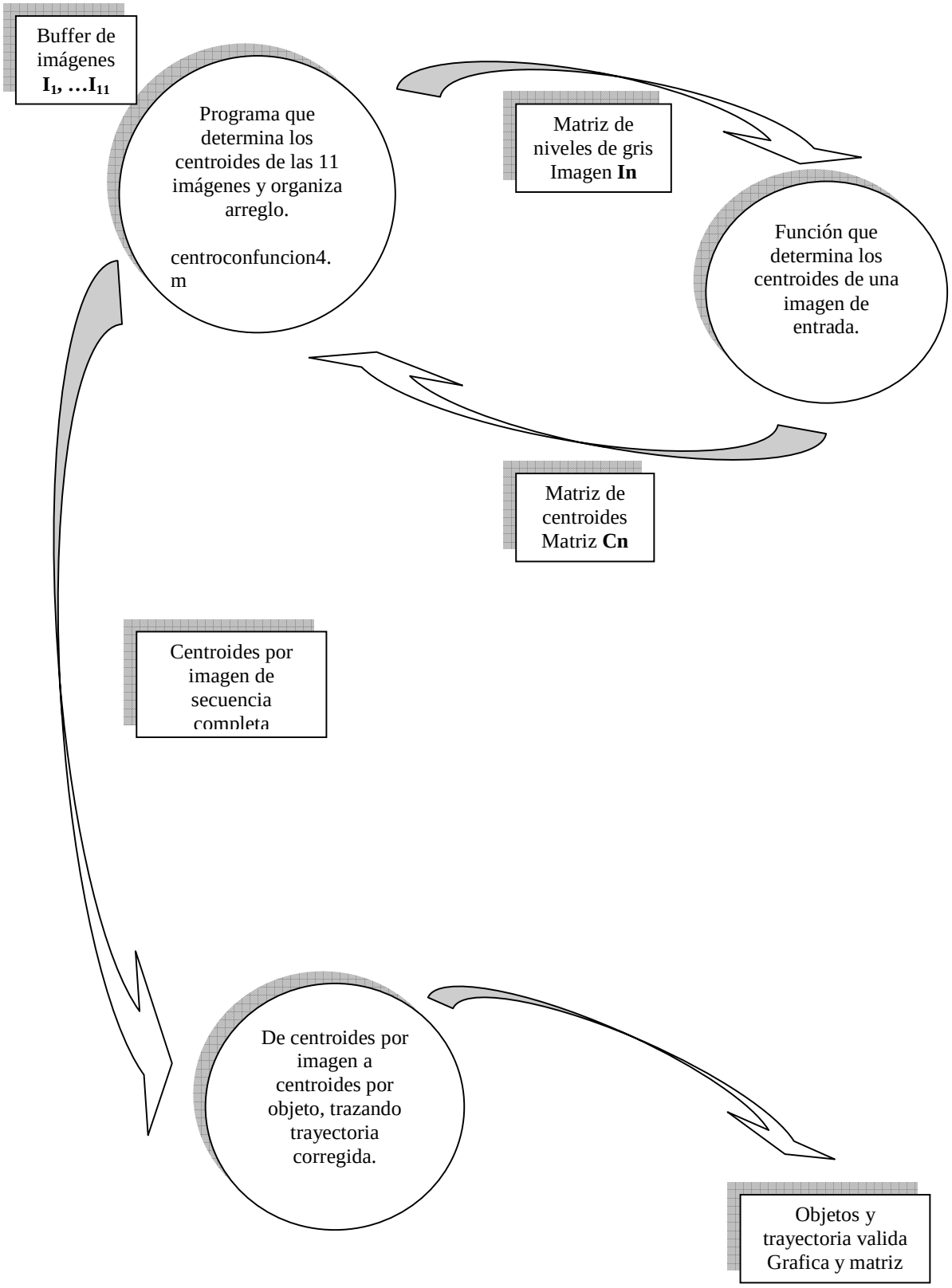
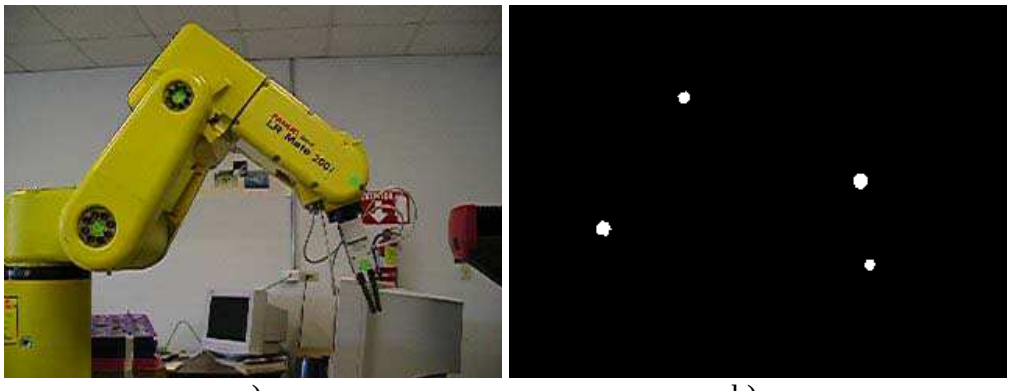


Figura 2.16. Secuencia final de los programas del seguimiento de la trayectoria de los objetos.

2.4 Preprocesamiento de secuencia de imágenes de brazo robot

Se utilizaron herramientas de Visión por Máquina para desarrollar un sistema de captura de video que permita medir trayectorias de movimiento de un brazo robótico. Una vez tomado el video y transferido a la computadora, el siguiente paso fue obtener una secuencia de imágenes con un submuestreo temporal a partir de la secuencia de video. Esto se realizó utilizando el software de conversión de formatos iVideoMAX 3.8 video, convirtiendo video a imágenes TIFF, logrando bajar de 15 cuadros por segundo a solo 3 cuadros por segundo. Con lo que se llega a una secuencia de baja resolución espacial y temporal, de sólo 85 imágenes que definen la trayectoria del brazo robótico que duraba 28.14 segundos aproximadamente, según la información de parámetros del video.

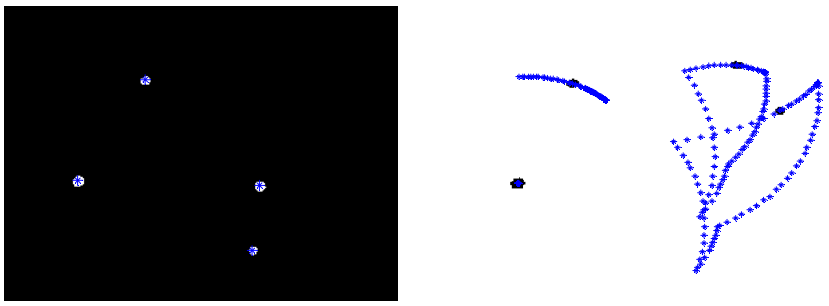
Para lograr el contraste en la escena de los puntos de interés seleccionados, con círculos verdes se marcaron 4 puntos de referencia que representan los 4 centros de los ejes del brazo robótica (figura 2.17a). El procesamiento de imágenes (segmentación) identifica estos puntos y los utiliza para representar los movimientos del robot. En todas las imágenes se obtuvieron los 4 puntos base indiscutiblemente, y todos los movimientos estuvieron contenidos en el campo visual del video, por lo que no hubo pérdida de información. Como punto estático de referencia espacial, se designó al primer eje ligado a la base del motor que solamente tuvo movimiento giratorio y no presento desplazamiento laterales programados; lo que nos permite determinar la incertidumbre debido al método visual, ya que las vibraciones del mecanismo es el único movimiento grabado de este eje.



a) b)

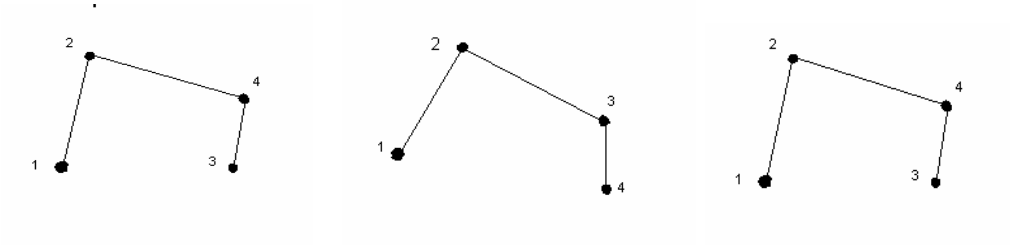
Figura 2.17. a) Brazo Robótico GE-Fanuc LR MATE 200i y b) su imagen binaria de los ejes del brazo de robot.

Como paso inicial para dejar las regiones de interés, en cada imagen se hace un procesado de color usando el software Photoshop, seleccionado un rango de color que corresponde a los círculos verdes y permitiendo un alto grado de difusividad de este color; con lo que se definen exactamente las regiones de referencia, para posteriormente crecerlas en un pixel y se elimina todo lo que se encuentra fuera. Subsecuentemente se les aplica un umbral de 50 y se convierte a modo de nivel de gris. Resultando imágenes en blanco y negro, donde aparecen claramente lo 4 puntos base (regiones) que representan la posición de los ejes (figura 2.17b). Esta imagen binaria es procesada después con Matlab utilizando la función *bwlabel*, obteniendo una imagen etiquetada y una variable que nos indica cuantas regiones conectadas existen en la imagen (objetos en la imagen). Una vez que la imagen ha sido etiquetada, se utiliza la instrucción *regionprops* para obtener el centroide, entre otros datos, de una región ya definida (objeto). Los datos de los centroides son las coordenadas de posición XY que en cada imagen se obtienen cuatro pares XY correspondientes a los ejes del robot. En la figura 2.18 se muestra la acumulación de todos los centroides encontrados en la secuencia de 85 imágenes. La cantidad de imágenes, y la de centroides por objeto son más que los encontrados con la secuencia de las hormiguitas (colores) de figura 2.11.



a) b)

Figura 2.18. a) Se muestran los centroides, marcados con una cruz azul, encontrados en la primera imagen, b) se muestra los centroides de los ejes de las 85 imágenes. Con los programas: *ejescentros.m* y *videorobotejes.m* respectivamente.



a) b) c)

Figura 2.19. Se muestra el posible cambio de etiquetas entre el eje 3 y eje 4, a) etiquetado inicial, b) intercambio de etiquetado y c) regreso al etiquetado inicial.

2.5 Seguimiento de trayectoria y análisis de movimiento de ejes de robot

Para la secuencia de imágenes del movimiento de un brazo de robot, el seguimiento de trayectoria de los ejes de rotación se hace a partir de las imágenes binarias que se encontraron el etiquetado de la función *bwlabel*. Después al aplicar la función *regionprops* se asigna una centroide a cada etiqueta, y pudiera parecer que ya se tiene la secuencia de objetos correctamente establecida. Pero como las funciones de Matlab trabajan con una sola imagen de entrada a la vez, no consideran la imagen anterior en la secuencia como etiqueta de referencia y puede producirse el intercambio de etiqueta automáticamente, como se muestre en el ejemplo de la figura 2.19.

La función *bwlabel* utiliza un orden de izquierda a derecha para numerar a como van apareciendo los objetos en la imagen. Y como la función utiliza una imagen entrada, no vincula las etiquetas con respecto a ninguna ejecución anterior, provocando el etiquetado erróneo para un mismo objeto de una secuencia de eventos en el tiempo.

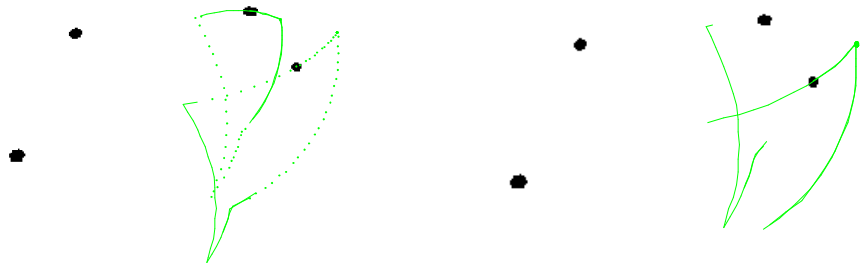
Si graficando la etiquetas y centroides, tal cual se generan para las 85 imágenes por las funciones *bwlabel y regioprops*, tendríamos la representación que se muestra en las figuras 2.20 y figura 2.21.

a) b)

Figura 2.20. El etiquetado simple resulta en la identificación correcta de la secuencia de movimiento de los ejes 1 y 2, a) trayectoria del primer eje, b) trayectoria del segundo eje.

Los cuatro puntos representan la posición inicial de los ejes en la secuencia, y la trayectoria se muestre en color verde.

La figura 2.21a muestra la trayectoria determinada agrupando los centroides del objeto etiquetado con el numero 3 durante toda la secuencia, y que debiera corresponder al eje 3, pero no sucede así. Se puede ver una disociación de trayectoria causada por el cambio de etiquetas de objetos al aplicar la función *BWLABEL* a cada imagen y agrupar estos objetos por etiqueta. La misma disociación se tiene con el eje 4 como se muestra en la figura 2.21b.



a) b)

Figura 2.21. Trayectoria del eje 3 y eje 4 determinada agrupando los objetos etiquetados como 3 (a) y 4 (b) respectivamente. La trayectoria se muestra con trazos de color verde, mientras que la posición inicial de los ejes se muestra con los puntos negros.

Para determinar la trayectoria verdadera de cada eje, se procede a implementar el mismo método de corrección que se utilizo en la secuencia de las hormigas; determinando la distancia mínima entre objetos en una secuencia consecutiva de imágenes sin importar la etiqueta del objeto. Y así ir agrupando centroides por objeto de acuerdo a la diferencia mínima sin importar su etiqueta. La figura 2.22 muestra la secuencia completa de proceso de determinación de secuencia de los ejes del robot. Como resultado se obtiene una matriz de centroides por eje. La figura 2.23 muestra el diagrama de flujo del programa para la determinación de trayectoria de los ejes de robot.

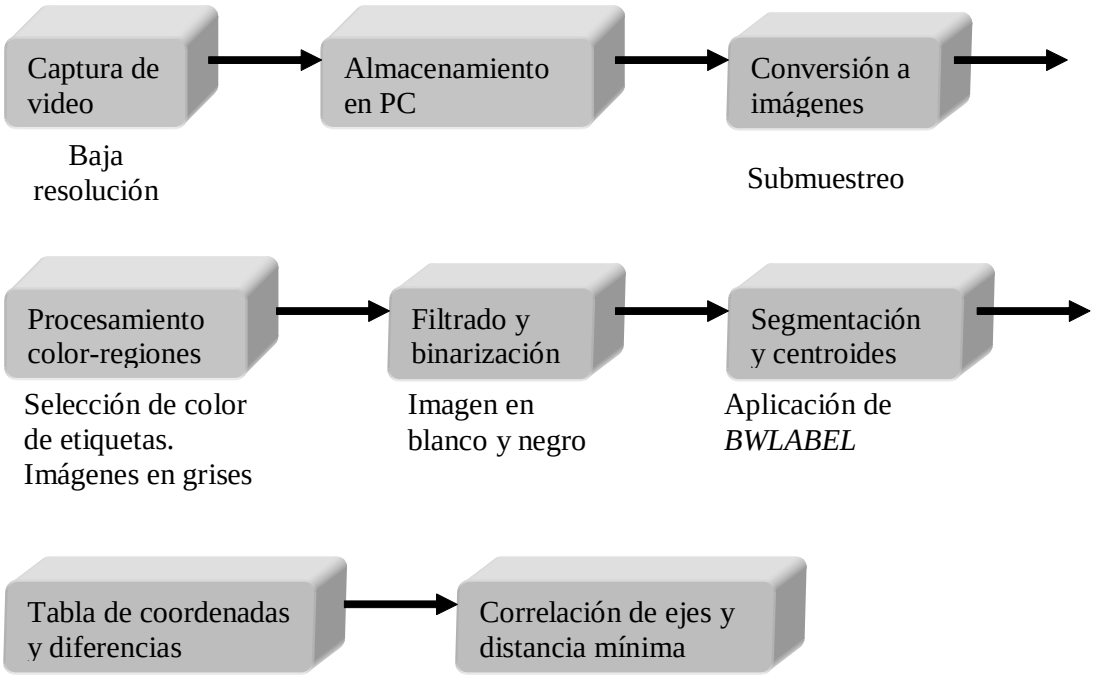


Figura 2.22. Continuidad del proceso de corrección de cruce de objetos.

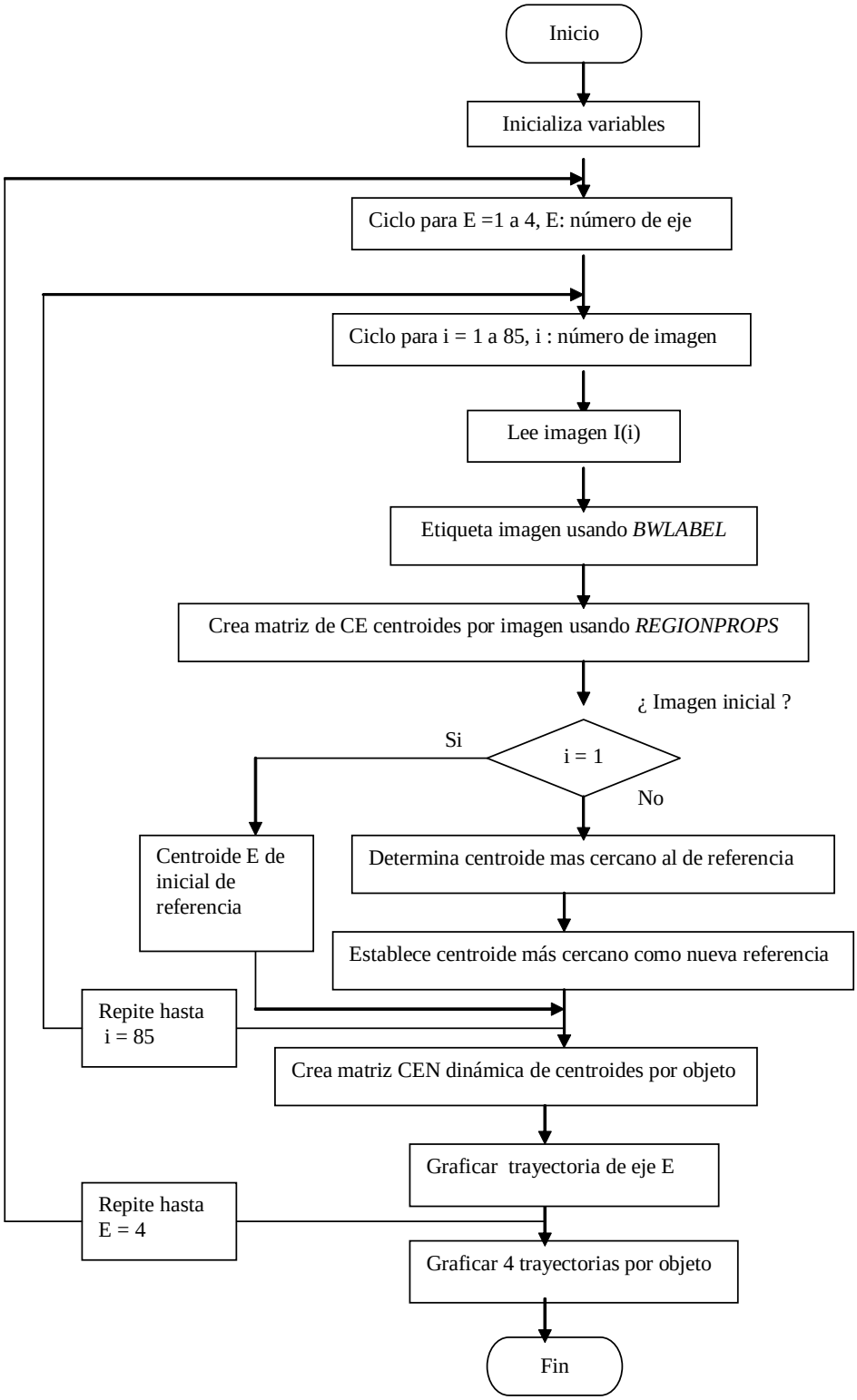


Figura 2.23. Diagrama de flujo para determinar las trayectorias de los ejes de un robot usando procedimiento de medir distancias entre centroides de imágenes consecutivas, asociando los centroides por objeto teniendo distancias mínimas entre ellos

Capítulo 3

Resultados y discusión

3.1 Trayectorias en la secuencia de imágenes de hormigas

Después de aplicar los métodos de filtrado, umbralizado y corrección de trayectorias entre imágenes de la secuencia, se obtiene las trayectorias verdaderas del movimiento de los objetos que aparecían en las imágenes (figura 3.1). Lo anterior resulta a pesar de que hubo pérdida de objetos (fuera de campo visual), de que las trayectorias no son uniformes y de que los objetos no son de área constante y geometría regular.

Como salida del proceso se obtiene una matriz de la secuencia de centroides por objeto, que es otro de los objetivos trazados, reducir la cantidad de información que tiene una imagen a una sencilla tabla. La tabla 3.1 a y b muestra estas coordenadas de los centroides.

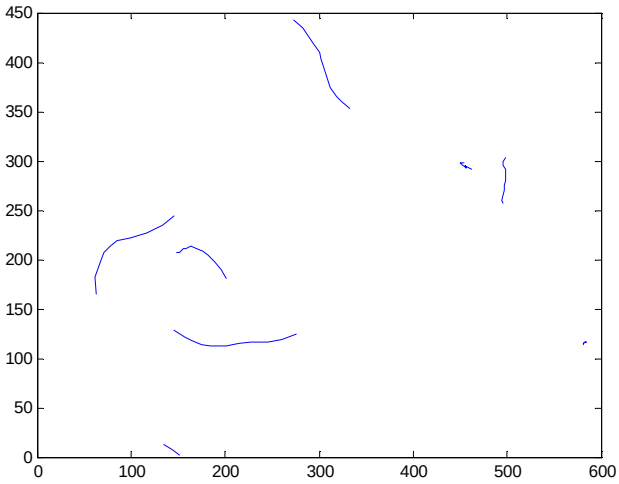


Figura 3.1. Se muestran las trayectorias reales de las hormigas, representada por los centroides de los objetos encontrados y corregidos en la secuencia de imágenes.

Tabla 3.1a. Coordenadas de los centroides por objeto, del 1 al 6

Objeto 1	
X	**Y**
63.25	165.77
61.40	182.15
67.46	197.11
71.04	206.74
77.92	213.69
85.11	219.62
98.44	221.50
116.89	227.13
125.71	231.31
132.77	235.01
146.16	244.48

Objeto 2	
X	**Y**
134.22	12.42
144.59	7.76
150.86	2.44

Objeto 3	
X	**Y**
146.38	128.86
156.34	121.38
165.57	117.46
175.00	113.35
184.78	112.99
200.92	113.10
215.01	114.73
227.93	115.88
244.74	116.96
261.19	119.48
275.49	123.86

Objeto 4	
X	**Y**
148.73	207.15
151.92	206.85
154.95	210.74
158.72	210.55
163.48	214.36
167.11	213.10
175.63	209.00
181.30	204.08
188.51	197.48
195.28	190.29
201.60	181.25

Objeto 5	
X	**Y**
332.14	352.89
324.15	359.48
318.21	365.54
311.41	374.17
306.26	391.37
302.06	403.63
301.14	410.19
293.56	419.22
285.68	430.51
282.32	434.75
273.02	442.91

Objeto 6	
X	**Y**
453.91	297.65
449.32	298.51
452.46	295.68
454.97	294.29
456.62	292.74
457.34	292.16
455.75	295.15
455.37	295.20
455.51	293.11
456.93	294.14
462.02	291.20

Tabla 3.1b. Coordenadas de los centroides por objeto, 7 y 8.

Objeto 7	
X	**Y**
498.20	303.41
495.44	299.25
495.16	295.77
498.26	291.73
498.42	286.14
497.77	279.83
497.23	276.12
496.59	269.86
496.16	264.95
494.35	259.59
495.13	256.68

Objeto 8	
X	**Y**
581.74	115.64
582.67	117.05
583.77	117.41
583.90	117.45
583.51	116.47
582.64	115.80
580.78	114.98
580.82	113.85
580.79	113.26
581.51	113.29
581.79	114.68

Durante el proceso final se pudo entender las condiciones o casos para poder aplicar las funciones aquí utilizadas y aunque son funciones que nos pueden realizar una tarea especifica de manera inmediata, cada una tiene parámetros que deben ser debidamente utilizados, como en el caso de determinar los centroides y el etiquetado de los objetos.

Para poder determinar las trayectorias en tiempo real no es posible realizarlo con Matlab, pero si con el mismo algoritmo. La desventaja de Matlab es el tiempo de ejecución y los ciclos que tiene que realizar para cada comando/instrucción/función.

Si la cantidad de objetos es tal que la distancia entre ellos es menor a su desplazamiento no se podrían determinar las trayectorias, ya que el etiquetado es secuencial a partir de distancias mínimas de centroides entre imágenes consecutivas.

3.2 Trayectoria de ejes de robot

El resultado de la determinación de la trayectoria de los ejes robot se muestra en la figura 3.2. La salida del programa *ejestrayectoria23.m* da la matriz de coordenadas corregidas que se grafican en esta figura.

Con los métodos empleados se resuelven las dificultades de convertir el video a graficas de trayectorias de los puntos de referencia. En la identificación de los ejes se obtiene una precisión satisfactoria de subpixel (<1mm) y se pudo observar cualitativamente la variación de la trayectoria que ocurre durante los ciclos de operación del robot

La tabla de centriodes por cada eje no se muestra, ya que es una secuencia extensa de 85 imágenes

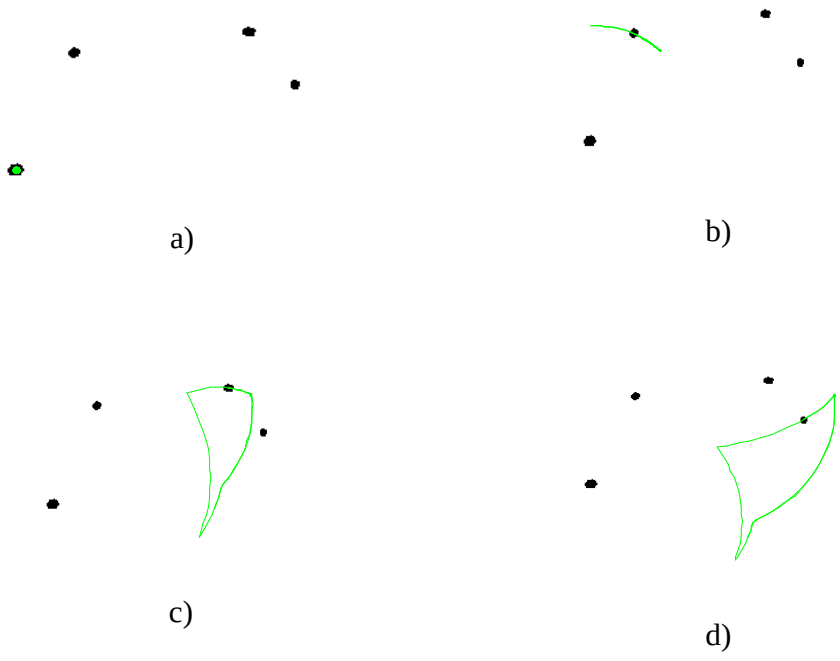


Figura 3.2. Trayectorias corregidas con el procedimiento de distancias mínimas, a) trayectoria del primer eje, b) segundo eje, c) tercer eje, d) cuarto eje.

Conclusiones

El uso de Matlab y sus funciones de procesamiento de imágenes estáticas limitan el diseño de programas cuando se quiere obtener características dinámicas a partir de una secuencia de imágenes. La implementación realizada incluye un paso de etiquetado por desplazamiento máximo, y sirve para la estimación del movimiento de objetos a partir de una secuencia de cuadros de video. El valor de desplazamiento máximo de objeto nos permite identificar los objetos que salen del campo de visión y que no haya trayectorias/desplazamientos que se asocien con otros objetos. El seguimiento del movimiento de los objetos es individual y está basado en un modelo de movimiento por segmentación de imágenes. Los objetos resultan en áreas amorfas de la imagen con características fotométricas homogéneas, concentrándonos en los centroides de estas áreas obteniendo coordenadas (x,y) para cada objeto de cada imagen, se puede sintetizar información de archivo de imágenes a grupo de datos en matriz que representa la trayectoria de cada objeto.

Este procedimiento fue aplicado a una secuencia de hormigas y una secuencia ejes de robot para medir secuencias de cuadros de video sin perder la identificación original. Aunque sucede alguna permutación durante el ciclo de operación (cruce de objetos) o perdida de objetos (salida del campo visual), se han obtenido trayectorias correctas de los movimientos de los objetos. El método presentado brinda una introducción a la estimación del movimiento en secuencia de imágenes. Y los resultados hasta el momento obtenidos nos incitan a seguir investigando en este campo hasta obtener los mejores resultados. Se tiene la posibilidad de avanzar más con la presente información, como por ejemplo la velocidad de desplazamiento, grado de movilidad, ángulos de dirección que pueden ser calculados a acuerdo al interés del movimiento. Como próximo paso se puede implementar sobre una secuencia en tiempo real.

El método puede integrarse sistemas de visión utilizado en diversos escenarios-objetos, con lo que se resuelven dificultades de requerir una cantidad masiva de memoria del video, a simples tablas y gráficas de trayectorias de datos coordenados de puntos de

referencia. Se obtiene a un bajo costo y sin la exigencia de recursos de alto desempeño, pues el método requiere de una diferencia simple para correlacionar la posición de un objeto sobre una secuencia temporal. Obteniendo así resultados favorables para un procedimiento en el seguimiento de objetos individuales y trazado de trayectoria con una herramienta de uso común reduciendo lo complejo de un sistema de inspección de seguimiento de trayectoria, y haciendo posible su automatización.

La selección cualitativa del filtro de imagen, la segmentación de grupos de pixeles que modelan áreas amorfas caracterizadas por un “centro de masa” o centroide, y el criterio de distancia máxima recorrida por cuadro, resultó en un algoritmo competente para el seguimiento de objetos, ya que nos permitió mostrar las trayectorias de movimiento de los objetos, aun y a pesar de la perdida de objetos en el campo visual y el intercambio secuencial de etiquetas. Además, el poder generar una matriz de centroides por objeto reduce la cantidad de información contenida en las secuencias de imágenes. Aunque la orientación y el área de los objetos no fue un parámetro necesario para determinar y caracterizar las trayectorias de los objetos, en el caso de que se hubieran usado nos darían más información sobre el comportamiento de los objetos, pero esto implica un mayor tiempo de procesamiento. Creemos que nuestro método ha logrado un balance entre tiempo de procesamiento contra datos y operaciones a realizar.

Bibliografía

[1] Aitzol Zuloaga Izaguirre, Jose Luis Martin Gonzalez, Luis Antonio Lopez Nozal, "**Determinación del movimiento a partir de secuencias de imágenes**"
Localización: Mundo electrónico, ISSN 0300-3787, Nº 278, 2, 1997 , pags. 26-30

[2] Al Bovik, 2009, **The Essential Guide to Image Processing**. CHAPTER 27 Computer-Assisted Microscopy; Fatima A. Merchant1 and Kenneth R. Castleman. ISBN: 978-0-12-374457-9.

[3] Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins. **Digital Image processing using MATLAB.** ISBN: 0130085197 , ISBN: 9780130085191 , Upper Saddle River, N.J. : Pearson Prentice Hall, c2004.

[4] R. García, J. Batlle, Ll. Magí, Ll. Pacheco, "**Seguimiento de Múltiples Objetos: un Enfoque Predictivo**", Revista Electrónica de Visión por Computador (REVC), no. 1, 2000.

[5] José Luis Martín, Aitzol Zuloaga, Josu Cruzado,"**Banco de pruebas para la estimación del movimiento basado en el cálculo del flujo óptico**"
XVI Simposium Nacional de la Unión Científica Internacional de Radio (URSI'01). Madrid (España), 2001.

[6] Alfonso Pérez, Hugo Solís, "**Métodos para seguimiento de dedo en tiempo real**"
http://www.dxarts.washington.edu/~hugosg/www/classes/seguidorDeDedo02.pdf
http://www.seccperu.org/files/Detecci%C3%B3ndeBordes-Canny.pdf

[7] Madasu Hanmandlu, Shantaram Vasikarla, Vamsi Krishna Madasu, "**Estimation of motion from a sequence of images using spherical projective geometry"**
P. Srimani, Proceedings of the International Conference on Information Technology: Computers and Communications. International Conference on Information Technology: Coding & Computing, Las Vegas, Nevada, (541-545). 28-30 April, 2003.

[8] David Edwards, Johnny T. Chang, Lin Shi, and Yizhou Yu; "**Motion Field Estimation for Temporal Textures**"; Department of Computer Science,University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA, Proceedings of the Seventh International Conference on Digital Image Computing: Techniques and Applications, DICTA 2003, 10-12 December 2003, Macquarie University, Sydney, Australia 2003.
http://www.cs.uiuc.edu/~yyz/publication/fme2.pdf

[9] Esqueda Elizondo, José Jaime, Luis Enrique Palafox Maestre, **Fundamentos de procesamiento de imágenes**, .
ISBN: 9707350164 , Mexicali, Baja California : Universidad Autónoma de Baja California, 2005.

[10] Marchand, Patrick. O. Thomas Holland. **Graphics and GUIs with MATLAB.** ISBN: 1584883200, 3a edición, Boca Raton, Fla. : Chapman & Hall/CRC, 2003.

[11] Jain, Ramesh, Rangachar Kasturi, Brian G. Schunck, **Machine vision** ISBN: 0070320187 (alk. paper) , ISBN: 0071134077 (pbk) Título: Machine vision / Ramesh Jain,. New York : McGraw-Hill, c1995,

[12] Pajares Martinsanz, Gonzalo, Jesús M. de la Cruz García., **Visión por computador : imágenes digitales y aplicaciones 1a ed.** , . ISBN: 9701508041 Edición: 1a ed. , México : Alfaomega, 2002.

Apéndices

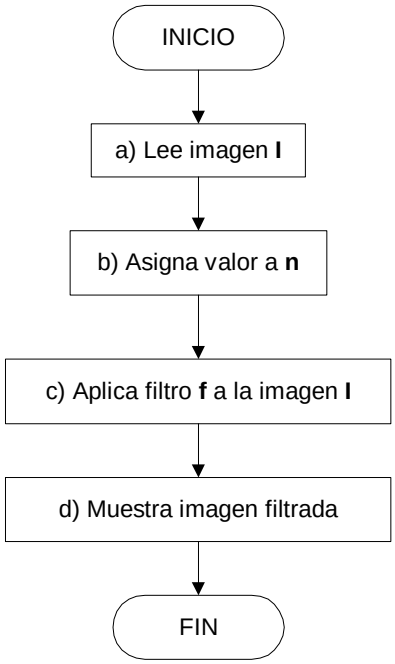
A) Filtrado de imagen I

Programa: filtromedhorm.m

Este programa filtra la imagen **I,** con un filtro de tipo mediano

filtromedhorm.m

Variable	Funcion
I	Matriz imagen en torno analizar
n	variable para dar el tamaño de matriz (n x n) de convolucion o mascara
f	Imagen filtrada, covolucionada



Salida: Imagen filtrada

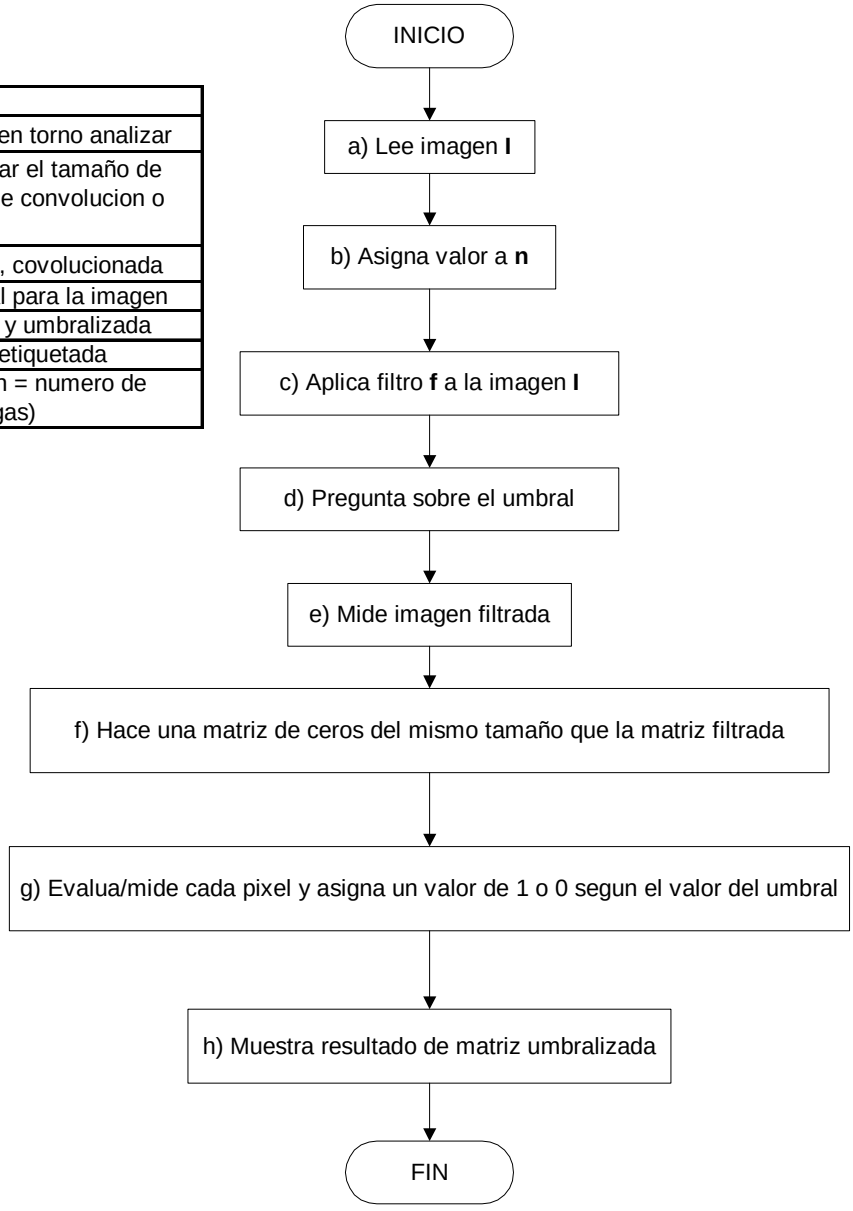
```
% Este programa filtra la imagen I, con un filtro de tipo mediano

    % a) Lee imagen I
I = imread('Anta08RD.tif');

    % b) Asigna valor a n
n=7;

    % c) Aplica filtro f a la imagen I
f = medfilt2(I, [n n], 'zeros');

    % d) Muestra imagen filtrada
imshow(f),title('f = medfilt2(I, [7 7], zeros)');
```

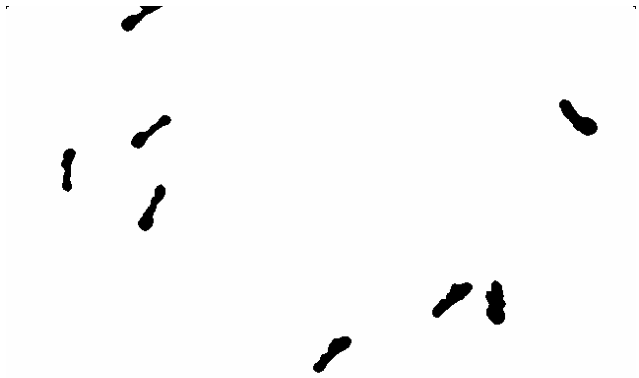
B) Resultado de imagen filtrada y umbralizada.

Este programa filtra la imagen **I** con un filtro de tipo mediano y aplica umbral

filtromedumbral.m

Variable	Funcion
I	Matriz imagen en torno analizar
n	variable para dar el tamaño de matriz (n x n) de convolucion o mascara
f	Imagen filtrada, covolucionada
t	valor de umbral para la imagen
Y	imagen filtrada y umbralizada
L	matriz imagen etiquetada
C	matriz (2 x h); h = numero de objetos (hormigas)





Salida: imagen umbralizada

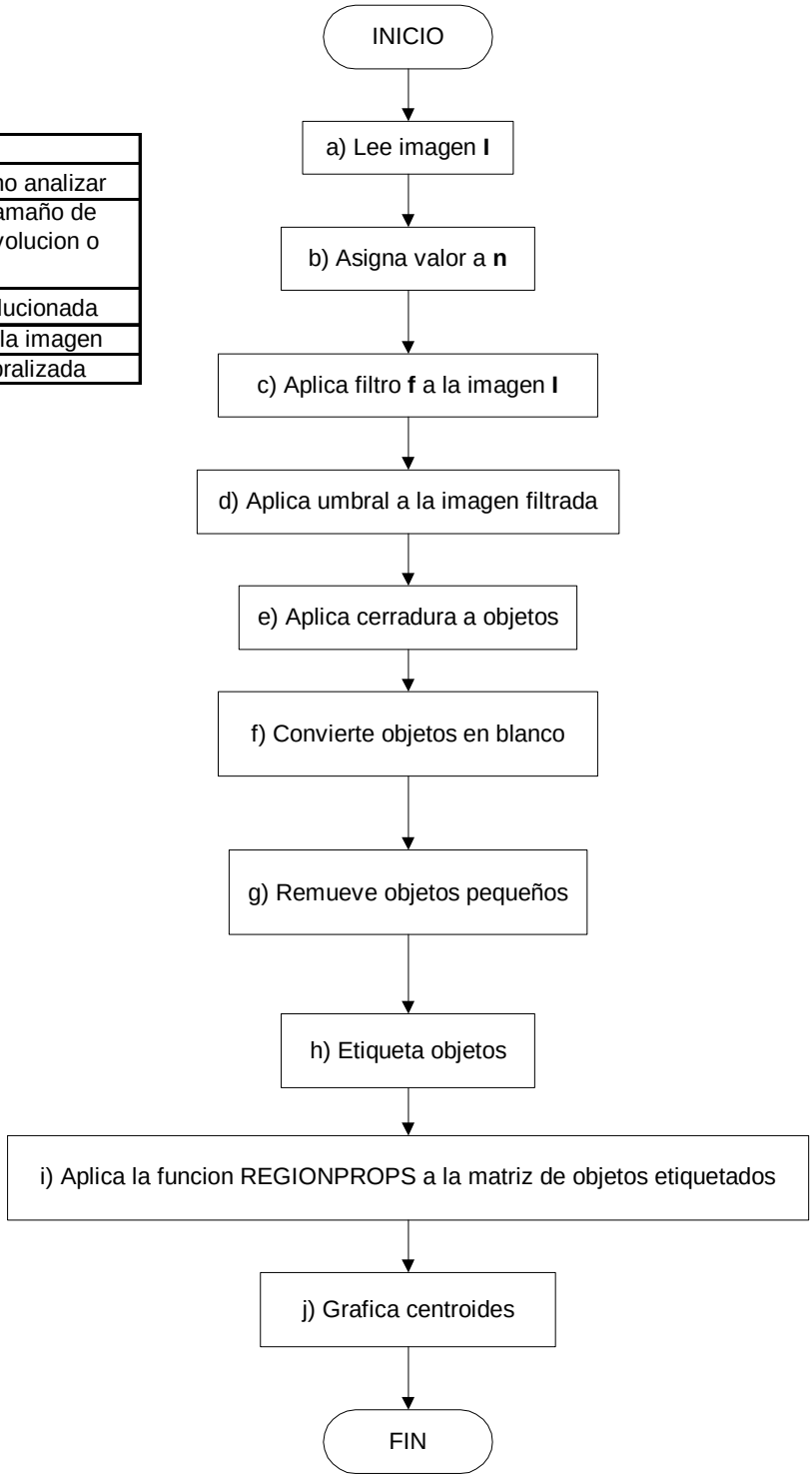
```
% Este programa filtra la imagen I, con un filtro de tipo mediano y aplica
% un umbral.

   % a) Lee imagen I
I = imread('Anta08RD.tif');
   % b) Asigna valor a n
n=7;
   % c) Aplica filtro f a la imagen I
g = medfilt2(I, [n n], 'zeros');
   % d) Pregunta sobre el umbral
t = input('umbral ? ');
   % e) Mide matriz o imagen filtrada
[N,M] = size(g);
   % f) Hace una matriz de ceros del mismo tamaño que la matriz filtrada
Y = zeros(N,M);
   % g) Evalua/mide cada pixel y asigna un valor de 1 o 0 segun el valor
   % del umbral a la matriz nueva
for i=1:N
  for j=1:M
    if (g(i,j) > t)
      Y(i,j) = 255;
    else
      Y(i,j) = 1;
    end
  end
end
   % h) Muestra resultado de matriz umbralizada
imshow(Y,gray(256));title('Imagen filtrada y umbralizada');
```

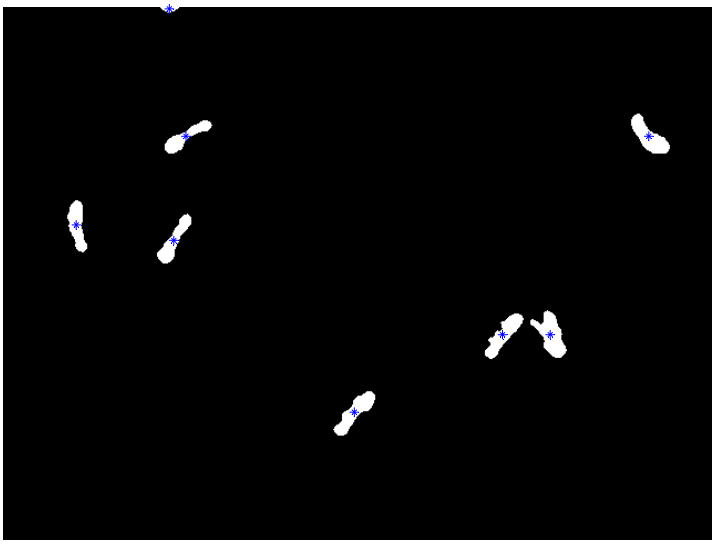
C) Muestra centroides de la primera imagen

Muestra centriodes de la primer imagen

closehormcentro.m

Variable	Funcion
I	Matriz imagen en torno analizar
n	variable para dar el tamaño de matriz (n x n) de convolucion o mascara
f	Imagen filtrada, covolucionada
t	valor de umbral para la imagen
Y	imagen filtrada y umbralizada





Salida : imagen con centroides graficados

```
% Muestra los centroides de los objetos

    % a) Lee imagen I
I = imread('Anta10RD.tif');
%imshow(I), title('Original image')

% Inicia proceso de filtrado FILTRO
    % b) Asigna valor a n, para el umbral
n=7;
    % c) Aplica filtro f a la imagen I
g = medfilt2(I, [n n], 'zeros');

% Inicia proceso de UMBRAL
%t = input('umbral ? ');
    % d) Aplica umbral a la imagen filtrada
t=100;
[N,M] = size(g);
Y = zeros(N,M);
for i=1:N
   for j=1:M
      if (g(i,j) > t)
         Y(i,j) = 1;    % hay que convertir a unos y ceros (imagen binaria) para usar las
funciones
      else              % imopen, imclose, imdilated, imerode, etc.
         Y(i,j) = 0;
```

```
    end
  end
end
%figure;
%imshow(Y);
   % e) Aplica cerradura a objetos para eliminar cualquier "imperfección"
   % dentro de las regiones (objetos/hormigas), elimina puntos negros dentro de las
regiones.

%CERRADURA
se = strel('disk', 1);
CY = imclose(Y, se);
%subplot(2,2,3);
   % f) Convierte objetos en blanco
NCY = ~CY;  % complemento de imagen CY (objetos negros-fondo blnaco), NCY (objetos
blancos)
        % para poder etiquetar objetos, BWLABEL co objetos blancos
figure;
imshow(NCY), title('closing the thresholded image');

   % g) Remueve objetos pequeños
% remove all object containing fewer than 30 pixels
bw = bwareaopen(NCY,30);

% fill a gap in the pen's cap
se = strel('disk',2);
bw = imclose(bw,se);

% fill any holes, so that regionprops can be used to estimate
% the area enclosed by each of the boundaries
bw = imfill(bw,'holes');
figure;
imshow(bw);


%APERTURA
%N=6
%se = strel('disk',N);
%se = strel('diamond',N);
%se = strel('octagon',N);
%se = strel('square',N);
%bw2 = imopen(CY,se);
%figure, imshow(bw2), title('After opening')

%C =~bw2;
   % h) Etiqueta objetos
```

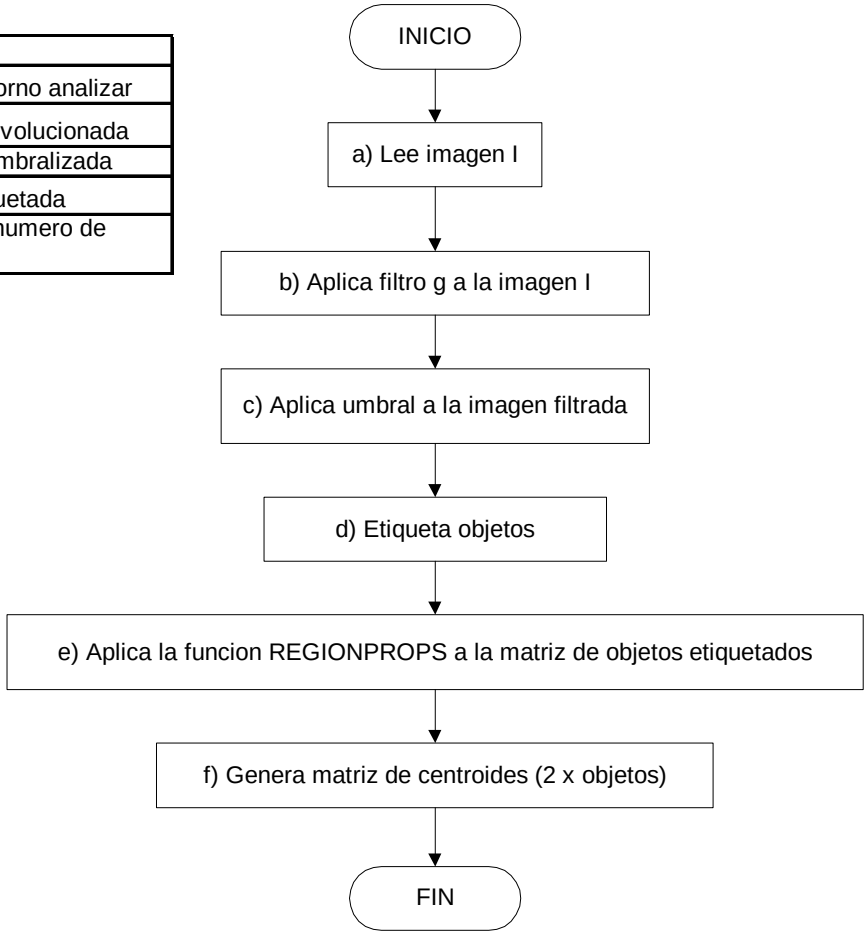
```
L = bwlabel(bw);            %L = BWLABEL(BW,N) returns a matrix L, of the same size as
BW, containing
    %labels for the connected components in BW. N can have a value of either
    %4 or 8, where 4 specifies 4-connected objects and 8 specifies
    %8-connected objects; if the argument is omitted, it defaults to 8.
    % i) Aplica la funcion REGIONPROPS a la matriz de objetos etiquetados
s  = regionprops(L,'Area', 'centroid');
centroids = cat(1, s.Centroid);     %CAT Concatenate arrays.
figure,imshow(bw)
hold on
    % j) Grafica centroides
plot(centroids(:,1), centroids(:,2), 'b*')
hold off
centroids    %muestra valores de centroides
%imwrite(bw,'centrohorm.tif')

%stats = regionprops(L,'Area','Centroid');

area = cat(1, s.Area);     %CAT Concatenate arrays.
area
```

D)

Este programa genera la funcion funcentro con entrada I (imagen) y salida la matriz 'centroids' de (2 x numero de objetos) ademas genera una imagen del area de las hormigas con un punto azul en el centro y el valor del area

funcentro2.m

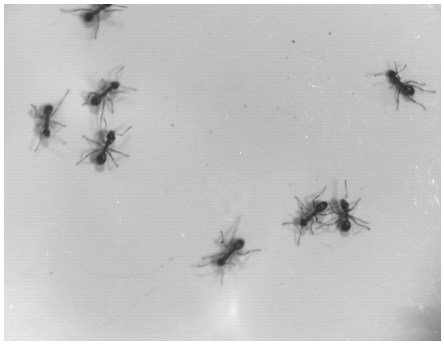
Variable	Funcion
E	Matriz imagen en torno analizar
g	Iimagen filtrada, covolucionada
Y	Imagen filtrada y umbralizada
L	Matriz imagen etiquetada
C	Matriz (2 x h); h = numero de objetos (hormigas)



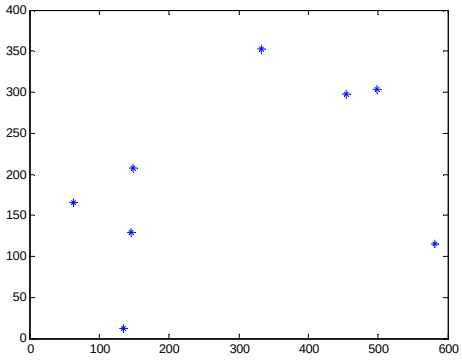
Entrada: imagen I

Salida: matriz centroides de (2 x objetos)
C1 =

```
  63.2461  165.7746
 134.2167   12.4151
 146.3753  128.8601
 148.7256  207.1545
 332.1439  352.8948
 453.9132  297.6528
 498.2042  303.4070
 581.7404  115.6417
```



Entrada: imagen I



Salida: grafica centroides

```
% Este programa genera la funcion funcentro con entrada I (imagen)
%y salida la matriz 'centroids' de 2 x numero de objetos
%ademas genera una imagen del area de las hormigas con un punto azul en
el
%centro y el valor del area

%function [output1,output2,...]=nombre(input1,input2,...)
function [centroids,bw] = funcentro(I)

%I = imread('ANTA11.tif');
%imshow(I), title('Original image')


    % a) Lee imagen I
E = I(1:475,1:635);
%figure;imshow(E);
%FILTRO
n=7;
    % b) Aplica filtro g a la imagen I
g = medfilt2(E, [n n], 'zeros');  %nueva imagen al aplicar filtro
%figure
%imshow(g),title('g = medfilt2(I, [7 7], zeros)')
```

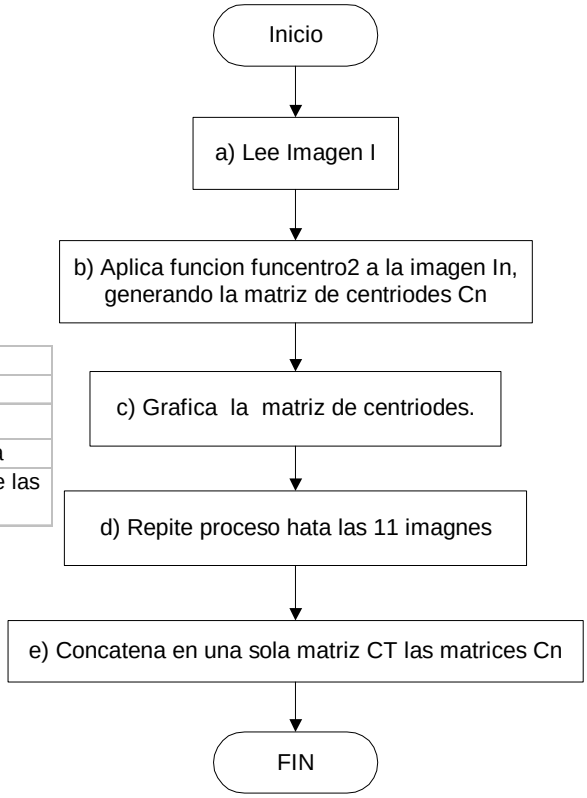
```
t=100;  % umbral de 100
    % c) Aplica umbral a la imagen filtrada
[N,M] = size(g);
Y = zeros(N,M);
for i=1:N
   for j=1:M
      if (g(i,j) > t)
         Y(i,j) = 1;    % hay que convertir a unos y ceros (imagen
binaria) para usar las funciones
      else              % imopen, imclose, imdilated, imerode, etc.
         Y(i,j) = 0;
      end
   end
end
%figure;
%imshow(Y);

%CERRADURA
se = strel('disk', 1);
CY = imclose(Y, se);  %hormigas negras
%subplot(2,2,3);

NCY = ~CY;  %complemento , hormigas blancas
%figure;
%imshow(NCY), title('closing the thresholded image, ~CY');

% remove all object containing fewer than 30 pixels
bw = bwareaopen(NCY,50);

% fill a gap in the pen's cap
se = strel('disk',2);
bw = imclose(bw,se);

% fill any holes, so that regionprops can be used to estimate
% the area enclosed by each of the boundaries
bw = imfill(bw,'holes');
%figure;
%imshow(bw);
%title('cerradura,sin ruido');

%este programa coloca un * en el centro de las hormigas
%realizado para entender las funciones de regionprops

    % d) Etiqueta objetos
L = bwlabel(bw);

%L = BWLABEL(BW,N) returns a matrix L, of the same size as BW, containing
    %labels for the connected components in BW. N can have a value of
either
    %4 or 8, where 4 specifies 4-connected objects and 8 specifies
    %8-connected objects; if the argument is omitted, it defaults to 8.

    % e) Aplica la funcion REGIONPROPS a la matriz de objetos etiquetados
s  = regionprops(L, 'centroid','Area');
```

```
    % f) Genera matriz de centroides (2 x objetos)
centroids = cat(1, s.Centroid);     %CAT Concatenate arrays.
%figure
%imshow(bw),title('centro,sin ruido');  %hormigas blancas , blanco (1),
negro (0)
%hold on
%plot(centroids(:,1), centroids(:,2), 'b*')  %grafica centros
%hold off
centroids;    %muestra valores de centroides
area = cat(1, s.Area);     %CAT Concatenate arrays.
area;
%hold on
%figure
%imagesc(bw);  %colorea las hormigas en tonos de gris
%for T = 1:length(s)  %escribe los valores de las area de cada hormiga
%text(s(T).Centroid(1)+15,s(T).Centroid(2)+15,num2str(s(T).Area),'color',
'yellow'); %extrae los elementos de la matriz 's'
%end

[labeled,numObjects] = bwlabel(bw,4);  %determina los objetos apartir de
bw

%numObjects

end  % END de funcentro

% VARIABLES

% E: matriz imagen en torno analizar

% g: imagen filtrada, covolucionada

% Y: imagen filtrada y umbralizada

% L: matriz imagen etiquetada

% C: matriz (2 x h); h = numero de objetos (hormigas)
```

E)

Este programa toma 11 imagnes, calcula los centroides de cada objeto en cada imagen, grafica los centroides trazando una trayectoria de cada objeto, usando la funcion (programa)funcentro2 el cual filtra, umbraliza, cierra los objetos, elimina impurezas y calcula centriodes

centroconfuncion4.m



Variable	Funcion
n	variable para numero de imagen
In	imagen en torno a analizar
Cn	Matriz centroide de imagen analizada
CT	Matriz que contiene los centriodes de las 11 imagenes

Salida: grafica de los centroides de las 11 imágenes

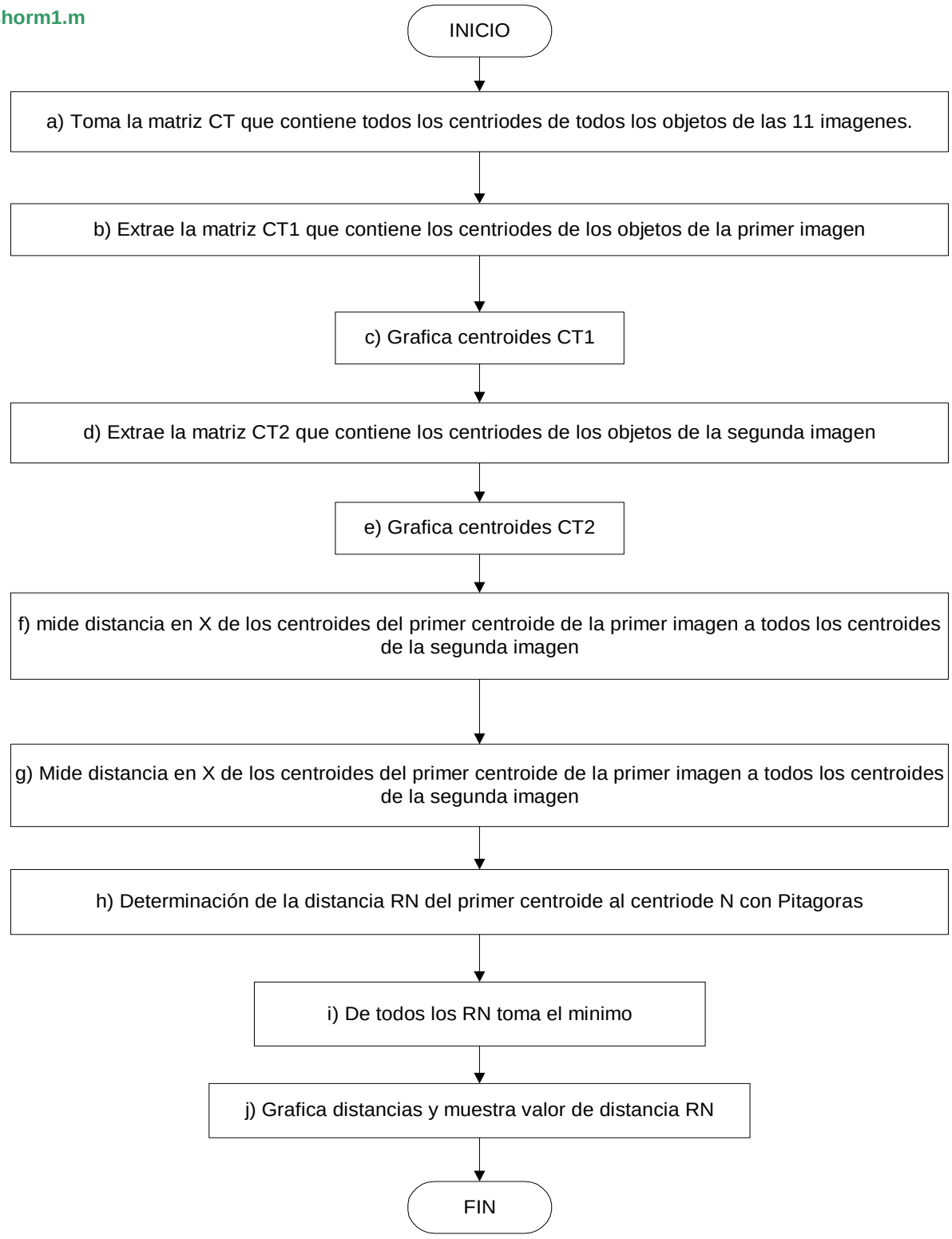
```
%a) lee imagen I
I1 = imread ('Anta08RD.tif');
%[output1,output2,...]=nombre(input1,input2,...)  , para usar la funcion
'funcentro'
%que calcula los centroides de las hormigas

% b) Aplica funcion funcentro2 a la imagen In, generando la matriz de
% centroides Cn
[C1,H]=funcentro2(I1);  % se obtienen los centroides de cada imagen
B=~H;  % B(hormigas negras), invierte H(hormigas blancas)
imshow(B);  %  hormigas negras
hold on
% c) Grafica  la  matriz de centroides.
plot(C1(:,1), C1(:,2), 'b*');  %grafica centros
hold on

% d) Repite proceso hata las 11 imagnes
I2 = imread ('Anta09RD.tif');
[C2]=funcentro2(I2);
plot(C2(:,1), C2(:,2), 'r*'); %grafica centros
hold on

I3 = imread ('Anta10RD.tif');
[C3]=funcentro2(I3);
plot(C3(:,1), C3(:,2), 'g*');  %grafica centros
hold on

I4 = imread('ANTA11.tif');
[C4]=funcentro2(I4);
plot(C4(:,1), C4(:,2), 'c*');  %grafica centros
hold on

I5 = imread ('ANTA12.tif');
[C5]=funcentro2(I5);
plot(C5(:,1), C5(:,2), 'y*');  %grafica centros
hold on

I6 = imread ('ANTA13.tif');
[C6]=funcentro2(I6);
plot(C6(:,1), C6(:,2), 'm*');  %grafica centros
hold on

I7 = imread ('ANTA14.tif');
[C7]=funcentro2(I7);
plot(C7(:,1), C7(:,2), 'k*');  %grafica centros
hold on

I8 = imread ('ANTA15.tif');
[C8]=funcentro2(I8);
plot(C8(:,1), C8(:,2), 'b*');  %grafica centros
hold on
```

```
I9 = imread ('ANTA16.tif');
[C9]=funcentro2(I9);
plot(C9(:,1), C9(:,2), 'r*');  %grafica centros
hold on

I10 = imread ('ANTA17.tif');
[C10]=funcentro2(I10);
plot(C10(:,1), C10(:,2), 'g*');  %grafica centros
hold on

I11 = imread ('ANTA18.tif');
[C11]=funcentro2(I11);
plot(C11(:,1), C11(:,2), 'c*');  %grafica centros
hold on

%e) Concatena en una sola matriz CT las matrices Cn
CT=cat(1,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11)  %concatenamos las matrices
de centroides

% VARIABLES

% n: numero de imagen

% In: imagen en torno a analizar

% Cn: matriz de centroides de la imagen In

% CT:   Matriz que contiene los centroides de las 11 imagenes
```

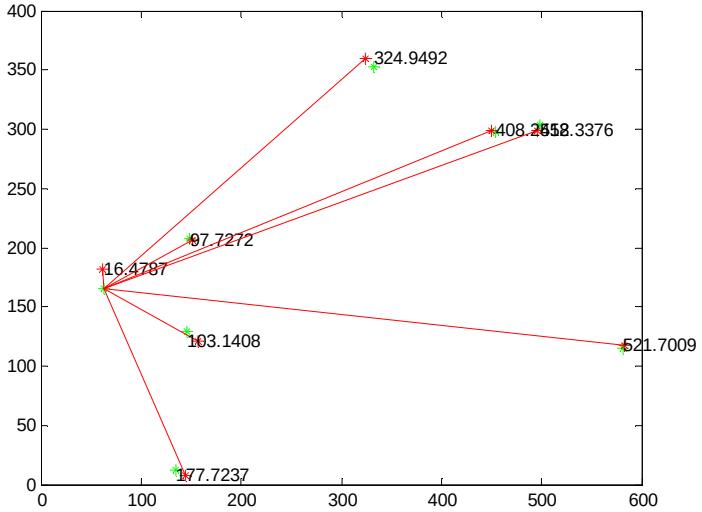
F)

Este programa mide y grafica las distancias del 1er objeto la todos de las siguientes imagenes

dishorm1.m



Variable	Función
K	Numero de objetos en la primer imagen
L	Numero de objetos en la segunda imagen
N	Numero de medición del primer objeto al objeto N
DX1(N)	Distancia en X de los centroides del primer centroide de la primer imagen a todos los centroides de la segunda imagen
DX2(N)	DX2(N): distancia en Y de los centroides del primer centroide de la primer imagen a todos los centroides de la segunda imagen
RN	RN: distancia absoluta del primer objeto al objeto N
M1	Valor mínimo de los RN
cs	Coordenadas del primer centroide
cd	Coordenadas del centroide al cual se esta midiendo



Proceso de medición.

```
% este programa mide y grafica las distancias del 1er objeto la todos de
% las siguientes imágenes, DF: DFdistamcia1; Capítulo 3/Seguimiento

   % a) Toma la matriz CT que contiene todos los centroides de todos los objetos de las 11
imágenes.

CT = [ ….]

   % b) Extrae la matriz CT1 que contiene los centroides de los objetos de la
   % primer imagen
CT1 =[63.2461 165.7746;  134.2167   12.4151;  146.3753  128.8601;  148.7256  207.1545;
332.1439  352.8948;  453.9132  297.6528;  498.2042  303.4070;  581.7404  115.6417 ];

   % c) Grafica centroides CT1
```

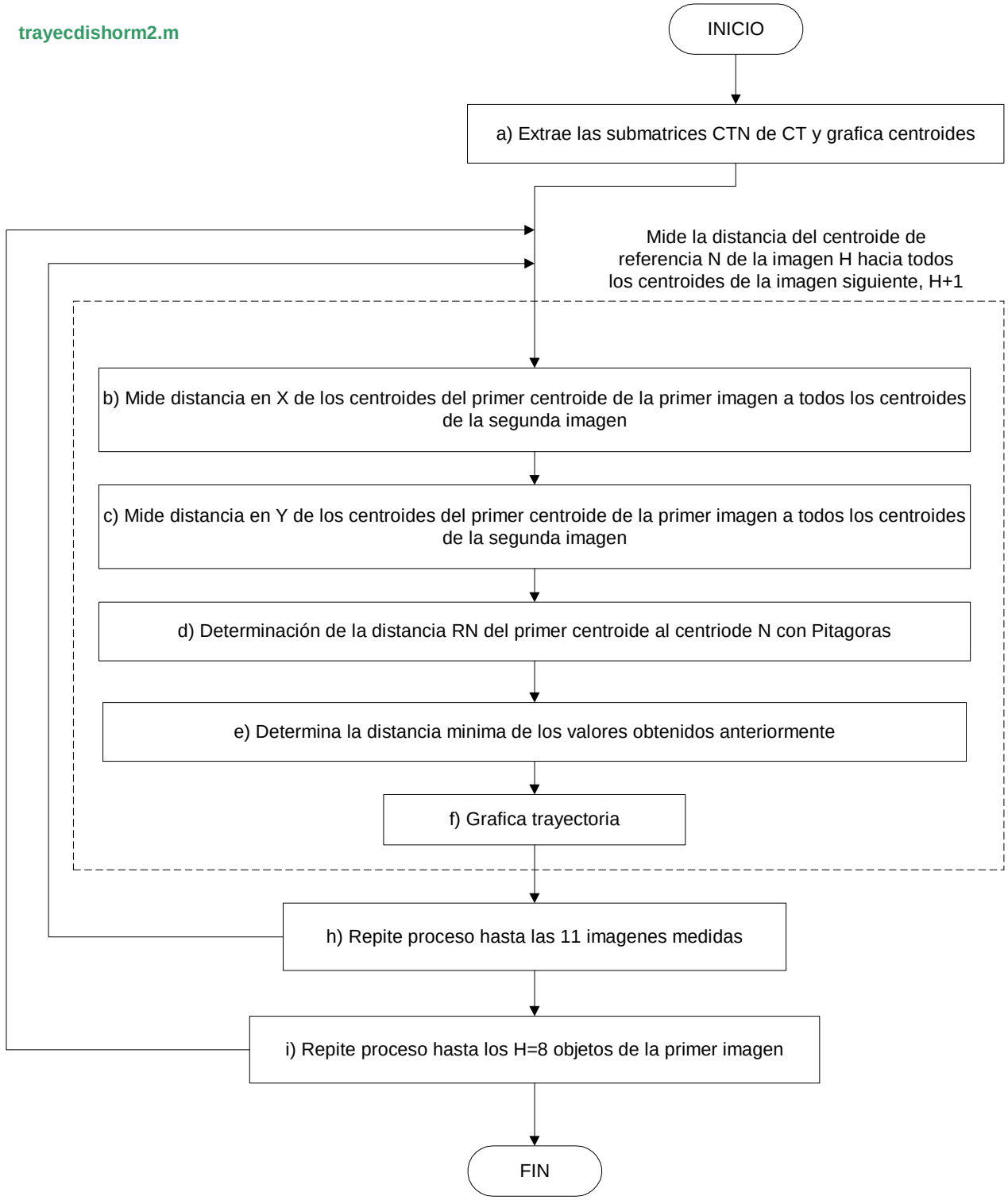
```
plot(CT1(:,1), CT1(:,2), 'g*');  %grafica imagen 1
hold on

   % d) Extrae la matriz CT2 que contiene los centroides de los objetos de la
   % segunda imagen
CT2 =[61.3973  182.1493;  144.5942   7.7613;  156.3423  121.3773;  151.9225  206.8489;
324.1496  359.4789;  449.3161  298.5129;  495.4424  299.2500;   582.6667 117.0504 ];

   % e) Grafica centroides CT2
plot(CT2(:,1), CT2(:,2), 'r*');  %grafica imagen 2

%1er medicion
K=length(CT1);
L=length(CT2);
for N=1:L    % primer punto de la primer imagen a todos los puntos de la segunda imagen
   % f) mide distancia en X de los centroides del primer centroide de la
   % primer imagen a todos los centroides de la segunda imagen
   DX1(N)=CT1(1)- CT2(N);
   % g) mide distancia en X de los centroides del primer centroide de la
   % primer imagen a todos los centroides de la segunda imagen
   DX2(N)=CT1(K+1)-CT2(L+N);
   % h) Determinación de la distancia RN del primer centroide al centroide N
   % con Pitagoras
   R1(N)=sqrt(DX1(N)^2 + DX2(N)^2);  %mide distancias
   % i) De todos los RN toma el minimo
   M1=min(R1);

   cs = [CT1(1),CT2(N)];
   cd = [CT1(K+1),CT2(L+N)];
   % j) grafica distancias y muestra valor de distancia RN
   plot(cs,cd,'r-');

   %text(61.3973,182.1493,num2str(R1(N)),'color','black');
   text(CT1(N),CT2(L+N),num2str(R1(N)),'color','black');

end

%for T = 1:L
%text(CT2(T),CT2(T+7),num2str(R1(T)),'color','black');
%end

% VARIABLES

% K: numero de objetos en la primer imagen
% L: numero de objetos en la segunda imagen
% N: numero de medicion del primer objeto al objeto N
% DX1(N): distancia en X de los centroides del primer centroide de la
   %primer imagen a todos los centroides de la segunda imagen
% DX2(N): distancia en Y de los centroides del primer centroide de la
   %primer imagen a todos los centroides de la segunda imagen
% RN: distancia absoluta del primer objeto al objeto N
```

```
%M1: valor minimo de los RN
% cs: coordenadas del primer centroide
% cd: coordenadas del centroide al cual se esta midiendo
```

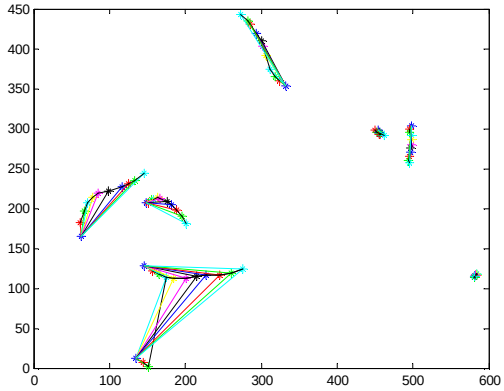
G)

Este programa mide distancias, determina distancias minimas, asocia centroides por objeto en una matriz y traza trayectorias.

trayecdishorm2.m



Variables	Funcion
CTN	submatrices de centroides por imagen
N	número de centroide de referencia
M	numero de medicion
IM	numero de imagen medida
MN	distancia minima de la medicion M
H	numero de objeto



Proceso de seguimiento

```
% Este programa mide distancias, determina distancias minimas, asocia
% centroides por objeto en una matriz y traza trayectorias.


   % a) Extrae las submatrices CTN de CT y grafica centroides

CT1 =[63.2461 165.7746;  134.2167   12.4151;  146.3753  128.8601;  148.7256  207.1545;  332.1439
352.8948;  453.9132  297.6528;  498.2042  303.4070;  581.7404  115.6417 ];
plot(CT1(:,1), CT1(:,2), 'b*');  %grafica imagen 1  (AZUL)
hold on

CT2 =[61.3973  182.1493;  144.5942   7.7613;  156.3423  121.3773;  151.9225  206.8489;  324.1496
359.4789;  449.3161  298.5129;  495.4424  299.2500;   582.6667 117.0504 ];
plot(CT2(:,1), CT2(:,2), 'r*');  %grafica imagen 2  (ROJO)

CT3 = [67.4586  197.1071;  154.9513  210.7424;  150.8596   2.4386;  165.5670  117.4554;  318.2059
365.5433;  452.4637  295.6832;  495.1638  295.7696;  583.7746  117.4057];
L3=length(CT3);
hold on
plot(CT3(:,1), CT3(:,2), 'g*');  %grafica imagen 3  (VERDE)

CT4=[71.0439  206.7405;  158.7159  210.5473;  175.0022  113.3488;  311.4143  374.1748;  454.9656
294.2899;  498.2627  291.7343;  583.8991  117.4462];
L4=length(CT4);
hold on
plot(CT4(:,1), CT4(:,2), 'c*');  %grafica 4ta imagen   (CYAN)
```

```
CT5=[77.9160  213.6914;  163.4769  214.3641;  184.7767  112.9901;  306.2627  391.3734;  456.6213
292.7351;  498.4198  286.1368;  583.5119  116.4693];
L5=length(CT5);
hold on
plot(CT5(:,1), CT5(:,2), 'y*');  %grafica 5ta imagen   (AMARILLO)

CT6 = [85.1101  219.6167;  167.1126  213.0954;  200.9209  113.1047;  302.0637  403.6336;  457.3448
292.1588;  497.7734  279.8293;  582.6405  115.8010];
L6=length(CT6);
hold on
plot(CT6(:,1), CT6(:,2), 'm*');  %grafica 6ta imagen   (MAGENTA)

CT7 = [98.4392  221.5039;  175.6310  209.0000;  215.0144  114.7344;  301.1355  410.1917;  455.7512
295.1459;  497.2334  276.1235;  580.7785  114.9818];
L7=length(CT7);
hold on
plot(CT7(:,1), CT7(:,2), 'k*');  %grafica 7ta imagen  (NEGRO)

CT8 = [116.8895  227.1341;  181.3027  204.0840;  227.9339  115.8833;  293.5558  419.2194;  455.3699
295.2000;  496.5930  269.8617;  580.8236  113.8528];
L8=length(CT8);
hold on
plot(CT8(:,1), CT8(:,2), 'b*');  %grafica 8ta imagen  (AZUL 2)

CT9 = [125.7068  231.3107;  188.5073  197.4822;  244.7391  116.9589;  285.6757  430.5109;  455.5091
293.1094;  496.1636  264.9470;  580.7876  113.2631];
L9=length(CT9);
hold on
plot(CT9(:,1), CT9(:,2), 'r*');  %grafica 9ta imagen  (ROJO 2)

CT10 = [132.7743  235.0124;  195.2752  190.2920;  261.1912  119.4835;  282.3151  434.7541;  456.9254
294.1410;  494.3514  259.5944;  581.5071  113.2857];
L10=length(CT10);
hold on
plot(CT10(:,1), CT10(:,2), 'g*');  %grafica 10ta imagen  (VERDE 2)

CT11 = [146.1581  244.4837;  201.5957  181.2516;  275.4880  123.8594;  273.0150  442.9135;  462.0178
291.2006;  495.1273  256.6788;  581.7893  114.6759];
L11=length(CT11);
hold on
plot(CT11(:,1), CT11(:,2), 'c*');  %grafica 11ta imagen   (CYAN 2)

% IM = 2:11
% Mide la distancia del centroide de referencia N de la imagen H
% hacia todos los centroides de la imagen siguiente, H+1
%for H=1:8    % numero de objeto
   H =1;
% 1ra medicion minima   (ROJO), M=1, IM=2
   K=length(CT1);
   L2=length(CT2);

   for N=1:L2    % primer punto de la primer imagen a todos los puntos de la segunda imagen
   % b) mide distancia en X de los centroides del primer centroide de la
   % primer imagen a todos los centroides de la segunda imagen
      DX1(N)=CT1(H)-CT2(N);
   % c) mide distancia en Y de los centroides del primer centroide de la
```

```
    % primer imagen a todos los centroides de la segunda imagen
       DX2(N)=CT1(K+H)-CT2(L2+N);
    % d) Determinación de la distancia RN del primer centroide al centroide N
    % con Pitagoras
       R1(N)=sqrt(DX1(N)^2 + DX2(N)^2);
     % De todos los RN toma el minimo
       M1=min(R1);  %calcula distancia minima 1
    % repite cubrir todos objetos
    end
    % e) Determina la distancia minima de los valores obtenidos
    % anteriormente
    for B1=1:L2
       if min(R1(B1))==M1;
          C1=B1;
       end
    end
    % f) Grafica trayectoria
    cs1 = [CT1(H),CT2(C1)];   %del 1er punto al mas cercano C1 de CT2
    cd1 = [CT1(K+H),CT2(L2+C1)];
    plot(cs1,cd1,'r-');    %grafica distancia minima 1 y grafica trayectoria 1
    % Nuevo centroide de referencia [ CT2(C1),CT2(L2+C1) ]

    % g) Repite proceso hasta las 11 imagenes medidas

% 2da medicion minima (VERDE), M=2 , IM=3
    for N=1:L3
    DX1(N)=CT2(C1)-CT3(N);

    DX2(N)=CT2(L2+C1)-CT3(L3+N);
    R2(N)=sqrt(DX1(N)^2 + DX2(N)^2);
    M2=min(R2);  %calcula distancia minima 2
    end

    for B2=1:L3   %
       if min(R2(B2))==M2;
          C2=B2;  % calcula numero de punto minimo de CT3
       end
    end

    cs2 = [CT1(H),CT3(C2)];   % x1,x3 del 1er punto al mas cercano C2 de CT2
    cd2 = [CT1(K+H),CT3(L3+C2)];  %y1,y3
    plot(cs2,cd2,'g-'); %grafica distancia minima 2

    cx = [CT2(C1),CT3(C2)];          % Xn-1, Xn
    cy = [CT2(L2+C1),CT3(L3+C2)];    % YN-1, Yn
    plot(cx,cy,'k-');   %grafica trayectoria 2

% 3ra medicion minima (CYAN), M=3, IM=4

    for N=1:L4
       DX1(N)=CT3(C2)-CT4(N); %distancia en X
       DX2(N)=CT3(L3+C2)-CT4(L4+N);   %distancia en Y
       R3(N)=sqrt(DX1(N)^2 + DX2(N)^2);  %mide distancias
       M3=min(R3);%calcula distancia minima 3
    end
```

```
    for B3=1:L4   %
       if min(R3(B3))==M3;
         C3=B3;
       end
    end

    cs3 = [CT1(H),CT4(C3)];      % x1,x4
    cd3 = [CT1(K+H),CT4(L4+C3)];  % y1,y4
    plot(cs3,cd3,'c-'); %grafica distancia minima 2

    cx = [CT3(C2),CT4(C3)];          % punto x3,x4   , Xn-1 , Xn
    cy = [CT3(L3+C2),CT4(L4+C3)];   % punto y3,x4     , Yn-1 , Yn
    plot(cx,cy,'k-');   %grafica trayectoria 2

% 4ra medicion minima (AMARILLO), M=4, IM=5

    for N=1:L5
       DX1(N)=CT4(C3)-CT5(N); %distancia en X
       DX2(N)=CT4(L4+C3)-CT5(L5+N);   %distancia en Y
       R4(N)=sqrt(DX1(N)^2 + DX2(N)^2);  %mide distancias
       M4=min(R4);%calcula distancia minima 4
    end

    for B4=1:L4   %
       if min(R4(B4))==M4;
         C4=B4;
       end
    end

    cs4 = [CT1(H),CT5(C4)]; % x1,x5
    cd4 = [CT1(K+H),CT5(L5+C4)];  % y1,y5
    plot(cs4,cd4,'y-'); %grafica distancia minima 4

    cx = [CT4(C3),CT5(C4)];
    cy = [CT4(L4+C3),CT5(L5+C4)];
    plot(cx,cy,'k-');   %grafica trayectoria 4

%5ta medicion minima (MAGENTA), M=5, IM=6

    for N=1:L6
    DX1(N)=CT5(C4)- CT6(N); %distancia en X
    DX2(N)=CT5(L5+C4)-CT6(L6+N);   %distancia en Y
    R5(N)=sqrt(DX1(N)^2 + DX2(N)^2);  %mide distancias
    M5=min(R5);
    end
    for B5=1:L5   %
       if min(R5(B5))==M5;
         C5=B5;
       end
    end
    cs4 = [CT1(H),CT6(C5)];     % x1,x6
    cd4 = [CT1(K+H),CT6(L6+C5)];   % y1,y6
    plot(cs4,cd4,'m-'); %grafica distancia minima 5

    cx = [CT5(C4),CT6(C5)];          %x5,x6
    cy = [CT5(L5+C4),CT6(L6+C5)];   %y5,y6
```

```
   plot(cx,cy,'k-');   %grafica trayectoria 5

%6ta medicion minima (NEGRO), M=6, IM=7

   for N=1:L7
   DX1(N)=CT6(C5)- CT7(N); %distancia en X
   DX2(N)=CT6(L6+C5)-CT7(L7+N);   %distancia en Y
   R6(N)=sqrt(DX1(N)^2 + DX2(N)^2);  %mide distancias
   M6=min(R6);
   end
   for B6=1:L6   %
      if min(R6(B6))==M6;
        C6=B6;
      end
   end
   cs4 = [CT1(H),CT7(C6)];     % x1,x7
   cd4 = [CT1(K+H),CT7(L7+C6)];   % y1,y7
   plot(cs4,cd4,'k-'); %grafica distancia minima 6

   cx = [CT6(C5),CT7(C6)];
   cy = [CT6(L6+C5),CT7(L7+C6)];
   plot(cx,cy,'k-');   %grafica trayectoria 6

%7ta medicion minima (AZUL 2), M=7, IM=8

   for N=1:L8
   DX1(N)=CT7(C6)- CT8(N); %distancia en X
   DX2(N)=CT7(L7+C6)-CT8(L8+N);   %distancia en Y
   R7(N)=sqrt(DX1(N)^2 + DX2(N)^2);  %mide distancias
   M7=min(R7);
   end
   for B7=1:L7   %
      if min(R7(B7))==M7;
        C7=B7;
      end
   end
   cs4 = [CT1(H),CT8(C7)];     % x1,x8
   cd4 = [CT1(K+H),CT8(L8+C7)];   % y1,y8
   plot(cs4,cd4,'b-'); %grafica distancia minima 7

   cx = [CT7(C6),CT8(C7)];
   cy = [CT7(L7+C6),CT8(L8+C7)];
   plot(cx,cy,'k-');   %grafica trayectoria 7

%8ta medicion minima (ROJO 2), M=8, IM=9

   for N=1:L9
   DX1(N)=CT8(C7)- CT9(N); %distancia en X
   DX2(N)=CT8(L8+C7)-CT9(L9+N);   %distancia en Y
   R8(N)=sqrt(DX1(N)^2 + DX2(N)^2);  %mide distancias
   M8=min(R8);
   end
   for B8=1:L8   %
      if min(R8(B8))==M8;
        C8=B8;
      end
```

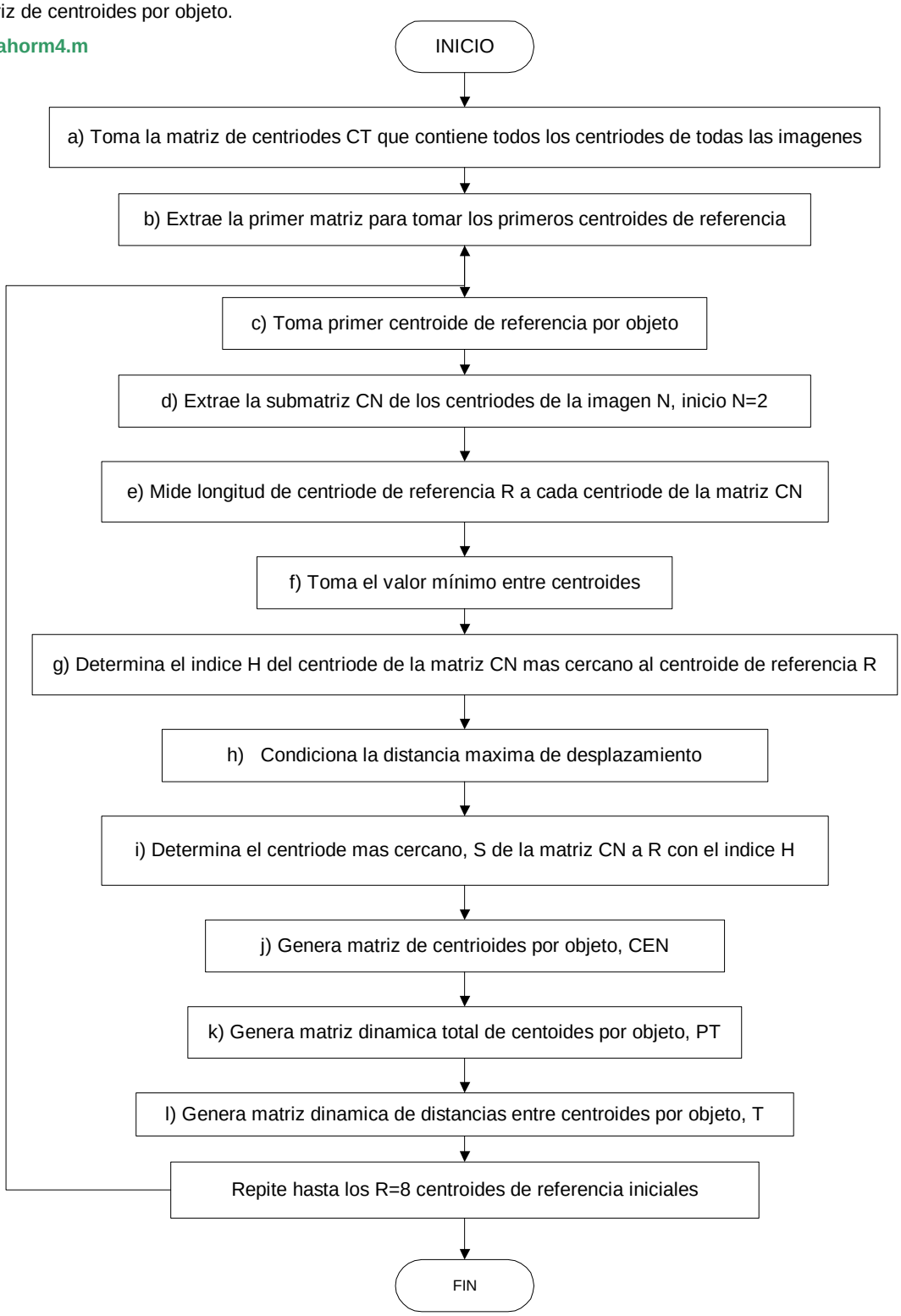
```
    end
    cs4 = [CT1(H),CT9(C8)];     % x1,x9
    cd4 = [CT1(K+H),CT9(L9+C8)];   % y1,y9
    plot(cs4,cd4,'r-'); %grafica distancia minima 7

    cx = [CT8(C7),CT9(C8)];
    cy = [CT8(L8+C7),CT9(L9+C8)];
    plot(cx,cy,'k-');   %grafica trayectoria 7

%9ta medicion minima (VERDE 2), M=9, IM=10

    for N=1:L10
    DX1(N)=CT9(C8)- CT10(N); %distancia en X
    DX2(N)=CT9(L9+C8)-CT10(L10+N);   %distancia en Y
    R9(N)=sqrt(DX1(N)^2 + DX2(N)^2);  %mide distancias
    M9=min(R9);
    end
    for B9=1:L9   %
       if min(R9(B9))==M9;
          C9=B9;
       end
    end
    cs4 = [CT1(H),CT10(C9)];     % x1,x10
    cd4 = [CT1(K+H),CT10(L10+C9)];   % y1,y10
    plot(cs4,cd4,'g-'); %grafica distancia minima 7

    cx = [CT9(C8),CT10(C9)];
    cy = [CT9(L9+C8),CT10(L10+C9)];
    plot(cx,cy,'k-');   %grafica trayectoria 7

%10ta medicion minima (CYAN 2), M=10, IM=11

    for N=1:L11
    DX1(N)=CT10(C9)- CT11(N); %distancia en X
    DX2(N)=CT10(L10+C9)-CT11(L11+N);   %distancia en Y
    R10(N)=sqrt(DX1(N)^2 + DX2(N)^2);  %mide distancias
    M10=min(R10);
    end
    for B10=1:L10   %
       if min(R10(B10))==M10;
          C10=B10;
       end
    end
    cs4 = [CT1(H),CT11(C10)];     % x1,x11
    cd4 = [CT1(K+H),CT11(L11+C10)];   % y1,y11
    plot(cs4,cd4,'c-'); %grafica distancia minima 7

    cx = [CT10(C9),CT11(C10)];
    cy = [CT10(L10+C9),CT11(L11+C10)];
    plot(cx,cy,'k-');   %grafica trayectoria 7

    % h) Repite proceso hasta los H=8 objetos de la primer imagen

%end

% VARIABLES
```

```
% CTN: submatrices de centroides por imagen
% N: número de centroide de referencia
% M: numero de medicion
% IM: numero de imagen medida
% MN: distancia minima de la medicion M
% H: numero de objeto
```

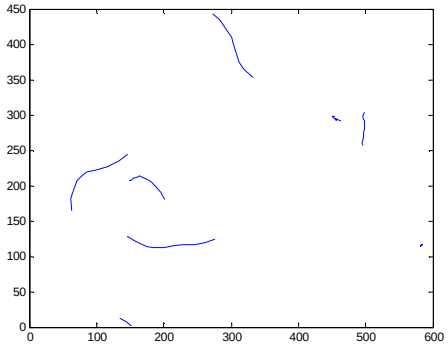
H) Cambiar el nombre del programa trayectoriahorm5.m

Este programa grafica la trayectoria de los objetos a travez de mediciones sucesivas entre los centriodes de cada imagen, determinando distancias minimas de centroides, teniendo como entrada matriz de centoides por imagen y salida matriz de centroides por objeto.

trayectoriahorm4.m



Variables	Función
R	Numero de centroide de referencia inicial por objeto
RF	Centroide de referencia inicial por objeto
N	Numero de imagen para medir distancias al centroide de referencia
DM	Valor mínimo entre centroides
PT	Matriz dinámica de N-centroides y distancias
T	Matriz dinámica de distancias de todos los centroides a su referencia.



Salida: trayectorias por objeto sin confusión

```
% Este programa agrupa los centroides por objeto y traza su trayectoria,
% analizando la distancia a centroides cercanos con la condicion de radio
% máximo igual a 20 pixeles
clear

% a) toma la matriz de centroides CT del programa "centroconfuncion4.m"

CT =[63.2461 165.7746;  134.2167   12.4151;  146.3753  128.8601;
148.7256  207.1545;  332.1439  352.8948;  453.9132  297.6528;  498.2042
303.4070;  581.7404  115.6417;   61.3973  182.1493;  144.5942    7.7613;
156.3423  121.3773;  151.9225  206.8489;  324.1496  359.4789;  449.3161
298.5129;  495.4424  299.2500;  582.6667  117.0504;   67.4586  197.1071;
154.9513  210.7424;  150.8596    2.4386;  165.5670  117.4554;  318.2059
365.5433;  452.4637  295.6832;  495.1638  295.7696;  583.7746  117.4057;
71.0439  206.7405;  158.7159  210.5473;  175.0022  113.3488;  311.4143
374.1748;  454.9656  294.2899;  498.2627  291.7343;  583.8991  117.4462;
77.9160  213.6914;  163.4769  214.3641;  184.7767  112.9901;  306.2627
391.3734;  456.6213  292.7351;  498.4198  286.1368;  583.5119  116.4693;
85.1101  219.6167;  167.1126  213.0954;  200.9209  113.1047;  302.0637
403.6336;  457.3448  292.1588;  497.7734  279.8293;  582.6405  115.8010;
98.4392  221.5039;  175.6310  209.0000;  215.0144  114.7344;  301.1355
410.1917;  455.7512  295.1459;  497.2334  276.1235;  580.7785  114.9818;
116.8895  227.1341;  181.3027  204.0840;  227.9339  115.8833;  293.5558
419.2194;  455.3699  295.2000;  496.5930  269.8617;  580.8236  113.8528;
125.7068  231.3107;  188.5073  197.4822;  244.7391  116.9589;  285.6757
430.5109;  455.5091  293.1094;  496.1636  264.9470;  580.7876  113.2631;
132.7743  235.0124;  195.2752  190.2920;  261.1912  119.4835;  282.3151
```

```
434.7541;  456.9254  294.1410;  494.3514  259.5944;  581.5071  113.2857;
146.1581  244.4837;  201.5957  181.2516;  275.4880  123.8594;  273.0150
442.9135;  462.0178  291.2006;  495.1273  256.6788;  581.7893  114.6759];
CT;  % muestra matriz centroides

%plot(CT(:,1), CT(:,2), 'g*');

L = [8 8 8 7 7 7 7 7 7 7 7];    %  numero de objetos (hotmigas) en las
imagenes

% b) extrae la primer matriz para tomar las primeras referencias
CR=CT(1:8,1:2);

% c) Toma primer centroide de referencia por objeto
% referencia
for R=1:L(1),      % este FOR es para medir, determinar y trazar las
trayectorias de los 8 objetos , L(1)
    R              %  muestra valor de R , numero de objeto

    %pause     % esta pausa muestra las trayectorias una por una

    RF=CT(R:R,1:2);   % extrae las referencias, Devuelve el elemento RF ;
x = R,1 , y = R,2

 %(B) el elemento R-esimo centroide (x,y)
    RFx=RF(1,1);    % valor en X de la referencia
    RFy=RF(1,2);    % valor en Y de la referencia
    %CEN

     CEN = 0; % inicializa a cero la matriz

    CEN(1,1)=RFx;    % primer elemento en X de centroides del objeto R
    CEN(1,2)=RFy;    % primer elemento en Y de centroides del objeto R

    %if R==1
        I=L(1)+1;     %  la variable I es para el indice de inicio para
extraer matriz
    %end
% d) Extrae la submatriz CN de los centroides de la imagen N, inicio N=2

    for N=2:11,

        if N==2
            RFx=RF(1,1);
            RFy=RF(1,2);
        else
            RFx=Sx;
            RFy=Sy;

        end
        LN=L(1:N);     % variable LN para ................
        I;    % indice de inicio para extraer matriz
        F=sum(LN);  % indice final para extraer matriz
```

```
        CN=CT(I:F,1:2);          % extrae la matriz de la imagen
consecutiva
        I=F+1;   % acarerea indice +1

        %  medicion minima (VERDE)

        C=L(N);    % lomgitud de la matriz de centroides
        length(CN);

        P=0;
% e) Mide longitud de centroide de referencia R a cada centroide
% de la matriz CN, poniendo los valores medidos en una matriz P

        for D=1:C     % ciclo para medir el punto de referencia RF a la
matriz CN

            DX1=RFx-CN(D);
            DX2=RFy-CN(D+C);
            P(D)=sqrt(DX1^2 + DX2^2);

% f) Toma el valor mínimo entre centroides, DM de P

            DM=min(P);  %calcula distancia minima 2,

        end

% g) Determina el indice H del centroide de la matriz CN mas cercano al
centroide de referencia R

        for B=1:D   %
            if min(P(B))==DM;
                DM;
                H=B;  % calcula numero (indice) de punto minimo de CT3
            end
        end
        DM;

 % h)   Condiciona la distancia maxima de desplazamiento

        if DM>20,break,end     % este IF corta el ciclo FOR que determina
los centroides
                               % cuando la distancia calculada es mayor
a
                               % 20 pixeles, pasando al siguiente objeto

% i) Determina el centroide mas cercano S de la matriz CN a R con el
indice H
        Sx=CN(H);         %
        Sy=CN(C+H);

        CEN(N,1)=Sx;    % primer elemento en X de centroides del objeto R
        CEN(N,2)=Sy;    % primer elemento en Y de centroides del objeto R
        CEN(N,3)=DM;
        MD(N)=DM;       % (H) toma como referencia
```

```
         %pause     % esta pausa muestra los centroides uno a uno y
distancia que se van calculando
                   % y anadiendo a la matriz de centroides

        %DM
    end

% j) Genera matriz de centrioides por objeto CEN
    CEN
    pause    % esta pausa va graficando trayectoria por objeto y matriz
centroides por objeto

    plot(CEN(:,1),CEN(:,2),'b-');
    hold on
% k) Genera matriz dinamica total de centoides por objeto, PT
    if  R==1
        PT=CEN;
        MTD=MD;   % matriz total de distancias

    end
% l) Genera matriz dinamica de distancias entre centroides por objeto, T
    if R>1
        F=[PT;CEN];
        %length(F)
        PT=F;

        T=[MTD;MD];
        MTD=T;

    end

end  % este END es de...........

PT % matriz de N-centroides y distancias.
T;  % matrz de distancias de todos los centroides a su referencia.

% el valor maximo de desplazamiento para una trayectoria normal es :
% 20 pixeles

% VARIABLES

% R: Numero de centroide de referencia inicial por objeto

% RF: Centroide de referencia inicial por objeto

% N: Numero de imagen para medir distancias al centroide de referencia

% DM: Valor minimo entre centroides

% PT: Matriz dinamica de N-centroides y distancias

% T: Matriz dinamica de distancias de todos los centroides a su
referencia.
```

I) TABLA DE CENTROIDES POR OBJETO

Esta tablas muestran los centroides ordenados por objeto (número de imagen, centroide asociado (x,y) y distancia entre centroides consecutivos), esta los datos de los centroides (x,y) son generados por el programa: **treyectoriahorm4.m**

Podemos observar que la tabla del objeto 2 del centroide 3 al 4 hay un incremento en distancia de pixeles de 105 y esto es por que el centroide (175.0022, 113.3488) en realidad pertenece al objeto 3, teniendo como resultado una confusión en seguimiento de trayectoria, que nos estaría mostrando el cambio repentino de lugar y dirección del objeto 2, siendo que en realidad el objeto del campo visual de la imagen 3 a la imagen 4.

Esto lo corregimos en el programa **treyectoriahorm5.m,** analizando los desplazamientos promedio.

Objeto 1 (Centroide)			
No. Imagen	Posición X	Posición Y	Incremento
1	63.2461	165.7746	0
2	61.3973	182.1493	16.4787
3	67.4586	197.1071	16.1392
4	71.0439	206.7405	10.2789
5	77.916	213.6914	9.7745
6	85.1101	219.6167	9.3201
7	98.4392	221.5039	13.462
8	116.8895	227.1341	19.2902
9	125.7068	231.3107	9.7565
10	132.7743	235.0124	7.9782
11	146.1581	244.4837	16.3961

Objeto 2			
Centroide	Posición X	Posición Y	Incremento
1	134.2167	12.4151	0
2	144.5942	7.7613	11.3732
3	150.8596	2.4386	8.2211
4	175.0022	113.3488	113.5074
5	184.7767	112.9901	9.7811
6	200.9209	113.1047	16.1446
7	215.0144	114.7344	14.1874
8	227.9339	115.8833	12.9705
9	244.7391	116.9589	16.8396
10	261.1912	119.4835	16.6447
11	275.488	123.8594	14.9515

Objeto 3			
Centroide	Posición X	Posición Y	Incremento
1	146.3753	128.8601	0
2	156.3423	121.3773	12.4633
3	165.567	117.4554	10.0238
4	175.0022	113.3488	10.2901
5	184.7767	112.9901	9.7811
6	200.9209	113.1047	16.1446
7	215.0144	114.7344	14.1874
8	227.9339	115.8833	12.9705
9	244.7391	116.9589	16.8396
10	261.1912	119.4835	16.6447
11	275.488	123.8594	14.9515

Objeto 4			
Centroide	Posición X	Posición Y	Incremento
1	148.7256	207.1545	0
2	151.9225	206.8489	3.2115
3	154.9513	210.7424	4.9328
4	158.7159	210.5473	3.7697
5	163.4769	214.3641	6.1021
6	167.1126	213.0954	3.8507
7	175.631	209	9.4517
8	181.3027	204.084	7.5057
9	188.5073	197.4822	9.7719
10	195.2752	190.292	9.8744
11	201.5957	181.2516	11.0308

Objeto 5			
Centroide	Posición X	Posición Y	Incremento
1	332.1439	352.8948	0
2	324.1496	359.4789	10.3566
3	318.2059	365.5433	8.4914
4	311.4143	374.1748	10.9831
5	306.2627	391.3734	17.9536
6	302.0637	403.6336	12.9593
7	301.1355	410.1917	6.6235
8	293.5558	419.2194	11.7878
9	285.6757	430.5109	13.7693
10	282.3151	434.7541	5.4128
11	273.015	442.9135	12.3721

Objeto 6			
Centroide	Posición X	Posición Y	Incremento
1	453.9132	297.6528	0
2	449.3161	298.5129	4.6769
3	452.4637	295.6832	4.2326
4	454.9656	294.2899	2.8637
5	456.6213	292.7351	2.2713
6	457.3448	292.1588	0.925
7	455.7512	295.1459	3.3856
8	455.3699	295.2	0.3851
9	455.5091	293.1094	2.0952
10	456.9254	294.141	1.7522
11	462.0178	291.2006	5.8803

Objeto 7			
Centroide	Posición X	Posición Y	Incremento
1	498.2042	303.407	0
2	495.4424	299.25	4.9908
3	495.1638	295.7696	3.4915
4	498.2627	291.7343	5.0879
5	498.4198	286.1368	5.5997
6	497.7734	279.8293	6.3405
7	497.2334	276.1235	3.7449
8	496.593	269.8617	6.2945
9	496.1636	264.947	4.9334
10	494.3514	259.5944	5.6511
11	495.1273	256.6788	3.0171

Objeto 8			
Centroide	Posición X	Posición Y	Incremento
1	581.7404	115.6417	0
2	582.6667	117.0504	1.686
3	583.7746	117.4057	1.1635
4	583.8991	117.4462	0.1309
5	583.5119	116.4693	1.0508
6	582.6405	115.801	1.0982
7	580.7785	114.9818	2.0342
8	580.8236	113.8528	1.1299
9	580.7876	113.2631	0.5908
10	581.5071	113.2857	0.7199
11	581.7893	114.6759	1.4186

J) Programa para determinar las trayectorias de los ejes del robot

```
% ESTE PROGRAMA MUESTRA LA TRAYECTORIA DE LOS EJES DE UN ROBOT

% INICIO

CEN=0;   % inicializa CEN, matriz de centriodes por objeto
C2=0;       % inicializa variable de indice numero de objeto de imagen

% A) Ciclo para determinar trayectoria del eje E, E:numero de eje
for E = 1:4   % ciclo FOR para numero de eje E de seguimineto de trayectoria

    A=1;      %  imagen A, inicial
    Z=85;       % imagen Z, final

% B) Ciclo para leer imagen y dedir distancias, i:numero de imagen
        for i=A:Z;   % ciclo FOR para leer imagen I(i) y tomar distancia minima y formar la matriz por
objeto (eje)

            I = imread(cat(2, 'ejes.', num2str(i,'%02d'),'.tif')); %  C) lee imagen i

            LA = bwlabel(I);  % D) etiqueta imagen I para aplicar funcion REGIONPROPS
            s  = regionprops(LA, 'centroid');   % E) aplica funcion REGIONPROPS para obtener centroides
            CE = cat(1, s.Centroid);     % F) CAT Concatenate arrays.  , CE: matriz de centroides por imagen

            if i==1   % aqui toma el primer punto de referencia E
       % G) Toma centroides de referencia iniciales del eje E
                Rx=CE(E);   % en X
                Ry=CE(4+E); % en Y
                                                        %IND=c+(E-1)*10
                CEN(1,1)=Rx;     % elemento de matriz columna en X para E
                CEN(1,2)=Ry;    % elemento de matriz columna en Y para E

                    % matriz (fila, columa)
                CEN(1,1)=Rx;   % asigna al primer elemento de la matriz por objetos el elemento de
referencia en x, matriz(fila, columna)
```

```
                CEN(1,2)=Ry;    % asigna al primer elemento de la matriz por objetos el elemento de
referencia en y
                CEN(1,3)=3;     % asigna al primer elemento de la matriz por objetos el elemento de
referencia etiqueta
                CEN(1,4)=1;     % columa 4 para numero de imagen
                CEN(1,5)=0;     % columna 5 para distancia minima
                CEN(1,6)=E;     %columna para eje determinado


            else

    %  H ) Mide distancias de centroides entre imágenes consecutivas,
    %  determina centroide mas cercano y establece este como nueva
    %  referencia
                L = 4;
                for N=1:L    % primer punto de la primer imagen a todos los puntos de la segunda imagen
                    DX1(N)=Rx - CE(N);            %DX1(N)=CI(E)- CP(N);
                    DX2(N)=Ry - CE(L+N);            %DX2(N)=CI(K+E)-CP(L+N);
                    R1(N)=sqrt(DX1(N)^2 + DX2(N)^2);  %mide distancias
                    M1=min(R1);     % toma la distancia minima

                end   % END del FOR N=1:L , para determinar distancia minima

      % I) El centroide mas cercano es el nuevo centroide de referencia
                for B2=1:L   %  ciclo para determinar indice de elemento minimo
                    if min(R1(B2))==M1;
                    C2=B2;  % calcula numero de punto minimo de CP
                    end   % END del IF min(R1(B2))==M1
                end    % END del FOR B2=1:L

                CEN(i,1)= CE(C2,1) ;    % columna de x
                CEN(i,2)= CE(C2,2) ;     % columna de y
                CEN(i,3)=C2;   %  columna de etiquetas
                CEN(i,4)=i;      % columna de imagen
                CEN(i,5)=M1;     % columna de distancia minima
                CEN(i,6)=E;
```

```
            Rx=CE(C2);   % el punto calculado en X pasa a ser la nueva referencia
            Ry=CE(L+C2); % el punto calculado en Y pasa a ser la nueva referencia
        end       %   END de IF i==1

    end     % END de FOR de leer imagen

  % Grafica trayectoria de eje E
    subplot(2,2,E);
    aux = imread('ejes.01.tif');
    imshow(~aux)
    hold on
    plot(CEN(:,1), CEN(:,2), 'r-');  %grafica centroides

  % Crea matriz dinámica de centroides por objeto
    if  E==1
        PT=CEN;
    end

    if E>1
        F=[PT;CEN];

        PT=F;
    end

end     % END de FOR de ejes

% grafica las trayectorias de todos los ejes
figure
aux = imread('ejes.01.tif');
imshow(~aux)
hold on
plot(PT(:,1), PT(:,2), 'r.');  %grafica centroides todos por objeto
```

```
% E : numero de eje
% i : numero de imagen
% I : imagen de entrada/lectura/analisis I(i)
% CE: matriz de centroides por imagen
% CEN : matriz de centriodes por objeto
% C2: :
```