Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Identificación de una configuración en un conjunto de puntos en el plano

Proyecto de investigación

José Daniel Faustinos Vargas
208333506
Van02_00@hotmail.com

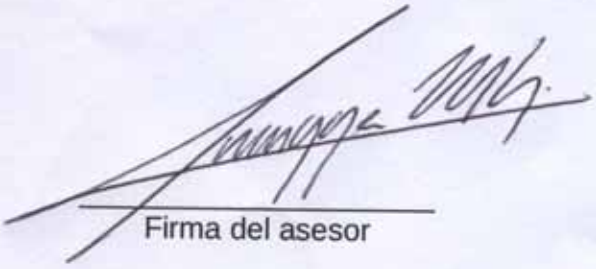
Trimestre 15 - I

Fecha de entrega:

Asesor: Francisco Javier Zaragoza Martínez
Profesor Titular
Departamento de Sistemas
franz@correo.azc.uam.mx

Asesor: Marco Antonio Heredia Velasco
Profesor Titular
Departamento de Sistemas
hvma@correo.azc.uam.mx

Yo, Francisco Javier Zaragoza Martínez, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



Firma del asesor

Yo, Marco Antonio Heredia Velasco, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Firma del asesor

Yo, José Daniel Faustinos Vargas, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

Firma del alumno

Resumen

Dado un conjunto de “k” puntos en el plano euclidiano, al cual llamaremos configuración, y un conjunto de “n” puntos también en el plano, al cual llamaremos espacio, se tiene el problema de decidir si la configuración de puntos se encuentra presente en el espacio, con “k” constante y n >= k.

En este proyecto se diseñan e implementan 4 algoritmos para las siguientes variantes de dicho problema:

- Buscar la configuración en el espacio de puntos sin modificación alguna.
- Buscar la configuración en el espacio de puntos permitiendo cambio de escala.
- Buscar la configuración en el espacio de puntos permitiendo rotación.
- Buscar la configuración en el espacio de puntos permitiendo rotación y escalamiento.

Cada algoritmo implementado nos dice si ésta presente la configuración que buscamos o no, y en caso de sí estar nos dice cuáles fueron los puntos del espacio correspondientes a los puntos de la configuración.

Tabla de contenido

Capítulo 1 “Introducción”

La falsificación de billetes es un problema muy serio, y con el desarrollo de nuevas tecnologías se ha venido incrementando a lo largo de los años. Por ello, el Banco de México [1] se ha visto en la necesidad de implementar medidas de seguridad [2], para que las personas puedan verificar la autenticidad de los billetes; marca de agua, hilo microimpreso, ventana transparente, verificación con luz negra, etc.

Curiosamente, los billetes cuentan con un elemento extra de seguridad, el cual no es mencionado por el Banco de México. Mientras jugaba con su fotocopiadora Xerox, la cual se negó a escanear un billete, Markus Kuhn [3] encontró un patrón de 5 círculos que venía en los billetes, que denominó EURion Constellation (figura 1), y el cual supuso era la razón por la que la fotocopiadora se negó a escanear el billete. Este mismo patrón puede observarse en billetes de varios países, ver figuras 2 y 3.

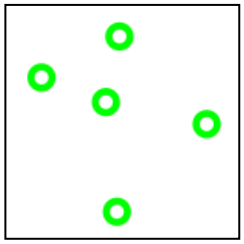


Figura 1 EURionConstellation

Figura 2 Billete de 10 euros

Figura 3 Billete de 20 dólares

Utilizando lo anterior como motivación, nos proponemos estudiar variantes del siguiente problema:

> *“Dada una configuración* C *de k puntos en el plano, decidir si* C *se encuentra presente en un conjunto* S *de n puntos en el plano, con k constante y n >= k”.*

Algunas variantes que pensamos estudiar son las siguientes:

- Buscar si en S se encuentra una copia de C sin modificación alguna.
- Buscar si en S se encuentra una copia de C, pero con diferente escala.
- Buscar si en S se encuentra una copia de C, pero con rotación.
- Buscar si en S se encuentra una copia de C, pero con rotación y diferente escala.

Capítulo 2 "Antecedentes"

Debido a su interés, la bibliografía sobre emparejamiento y reconocimiento de patrones es vasta y resultaría imposible citar todos los trabajos relacionados a nuestro problema, por lo cual sólo mencionaremos algunos que nos parecieron de especial interés.

2.1 Internos

Dentro de la UAM Azcapotzalco se han trabajado proyectos terminales similares, como lo son:

- "Algoritmo de búsqueda de configuraciones de enlaces ortogonales en dos y tres dimensiones" propuesto en el trimestre 10-P [5]. Este proyecto señala las diferentes formas abiertas y cerradas que se puede tener de un alambre articulado en dos y tres dimensiones, con enlaces unitarios y dobles, pero no señala si se encuentra o no una configuración de puntos especifico.

- "Separación automática de fondo y fenómeno de interés en imágenes tipo" propuesto en el trimestre 13-P [6]. Este proyecto, se encarga que dada una imagen (figuras en cultivos) obtiene como resultado, el contorno de la figura formada en dicha imagen, no da información alguna si hay una cierta configuración asume que debe haber una figura y sobre ella trabaja.

2.2 Externos

- "Shih-Hsu Chang, Fang-Hsuan Cheng, Wen-HsingHsu Guo-Zua Wu" [7]. Estas personas resuelven un problema similar, el cual dadas dos configuraciones de puntos hacen la comparación de ellas con las características de que pueden estar rotadas o tener un cambio de escala.

- "Lihua Zhang, WenliXu, Cheng Chang" [8].Estas personas resuelven un problema similar, el cual es que dadas dos configuraciones de puntos, ya sea que se encuentren rotadas o con un cambio de escala hacen la comparación de ellas utilizando matrices.

- "OswinAichholzer, Ruy Fabila-Monroy, Hernán González-Aguilar, Thomas Hackl, Marco A. Heredia, Clemens Huemer, Jorge Urrutia, BirgitVogtenhuber" [9]. Estas personas se proponen contar, de un espacio de n puntos, todos los cuadriláteros que allá, no importando si los cuadriláteros comparten puntos. Dan cotas inferiores y superiores sobre este número de cuadriláteros.

Capítulo 3 “Justificación”

Aunque el problema de encontrar la EURion Constellation dentro de un billete es interesante por sí mismo, no lo estudiaremos directamente por diversas razones: porque los detalles alrededor de la EURion Constellation se mantienen en secreto por parte de los bancos; porque está patentado [4] el proceso de identificación de este patrón por medios electrónicos; por la dificultad propia de buscar patrones dentro de una imagen. En su lugar trabajaremos en un problema simplificado, buscando puntos en un conjunto, en lugar de círculos en una imagen; y generalizaremos el problema al buscar cualquier configuración de *k* puntos, en lugar de sólo un conjunto de 5 puntos.

Sobra decir que, gracias a la generalidad del problema que vamos a estudiar, podrá ser usado como base para el desarrollo futuro de otras aplicaciones relacionadas con búsqueda de patrones, y si se deseara, podría usarse para el problema específico de la EURion Constellation.

Capítulo 4 "Objetivos"

4.1 Objetivo general

Diseñar e implementar algoritmos que decidan si una cierta configuración de puntos está presente en el espacio de puntos en el plano.

4.2 Objetivos específicos

- Diseñar e implementar un algoritmo que busque la configuración de puntos sin modificaciones.
- Diseñar e implementar un algoritmo que busque la configuración de puntos permitiendo cambio de escala.
- Diseñar e implementar un algoritmo que busque la configuración de puntos permitiendo rotación.
- Diseñar e implementar un algoritmo que busque la configuración de puntos permitiendo rotación y escalamiento.

Hacer un análisis de los algoritmos diseñados.

Capítulo 5 “Marco teórico”

Tanto el problema que se busca resolver en este proyecto, como la identificación de la EURion Constellation en billetes, están dentro de los temas investigados por el área de reconocimiento de patrones.

5.1 Reconocimiento de patrones.

El reconocimiento de patrones es la ciencia de la computación que se encarga del reconocimiento de objetos, personas, señales, representaciones, etc., en un espacio determinado.

Estos son algunos ejemplos de reconocimiento de patrones:

Smile Shooter
Lleva acabo dos tareas:

Enfocarse en un rostro dentro de una imagen.

Una vez detectado el rostro, se busca una sonrisa genuina es decir que busca en una base de datos si eso corresponde a alguna sonrisa.

Este proceso se hace cuando se va a tomar la foto.

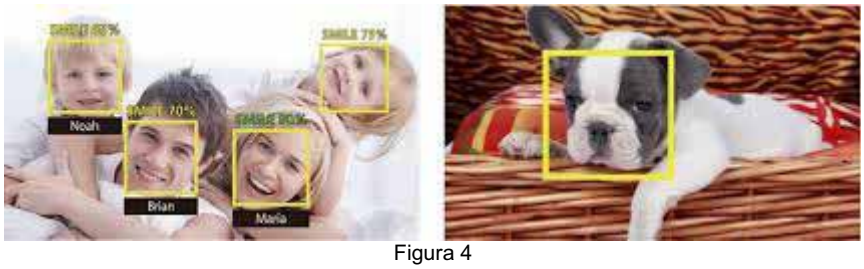


Figura 4

Kinect
El kinect funciona de la siguiente manera con una cámara de profundidad hace lo siguiente: captura los movimientos de las personas utilizando la tecnología de una cámara RGB y con la cámara de infrarrojos diferencia la profundidad.

Figura 5 Kinect

Lector de huella digital

Lleva a cabo dos tareas:

1) Obtener una imagen de su huella digital, y
2) Comparar el patrón de valles y crestas de dicha imagen con los patrones de las huellas que tiene almacenadas.

Los dos métodos principales de obtener una imagen de una huella digital son por lectura óptica o lectura de capacitancia.

Figura 6 Detector de huella

5.2 Conceptos.

Para poder comprender de forma adecuada lo que se realizó en este proyecto, hay que familiarizarse con algunos conceptos relacionados con las áreas de análisis de algoritmos y reconocimiento de patrones.

5.2.1 Transformaciones afines.

Un concepto clave en el reconocimiento de patrones entre conjuntos de puntos son las transformaciones a fines, que serán definidas a través de los siguientes conceptos:

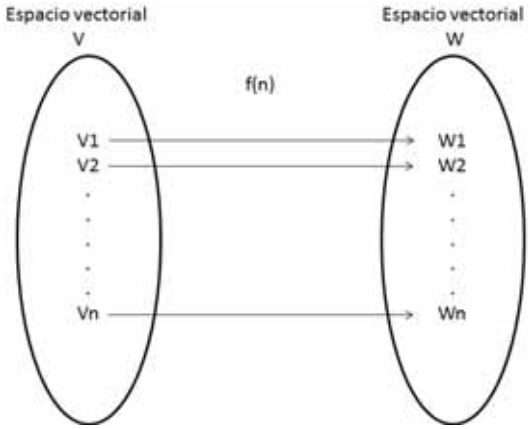
Punto: Es una figura geométrica adimensional, “no tiene longitud, área, volumen, ni otro ángulo dimensional”. Describe una posición en el espacio, determinada respecto de un sistema de coordenadas preestablecidas.

Distancia: Es la longitud del segmento de la recta que los une, expresado numéricamente.

Angulo: Es la parte del plano comprendida entre dos semirrectas que tienen el mismo punto de origen o vértice (en este proyecto utilizaremos el eje x como referencia).

Una transformación afín en geometría es una “transformación lineal + translación”, para comprender se definirán los conceptos:

Transformación lineal: Es una función f entre espacios vectoriales ,es decir transforma un espacio vectorial a otro, para señalar una transformación lineal se usa f(V) = W donde V y W son los espacios vectoriales que actúan dentro de un mismo campo, para entender mejor se muestra en la siguiente figura:



Traslación: Se define como movimientos directos sin cambios de forma y tamaño de las figuras u objetos a los cuales se desplazan según el vector.

Transformación afín: En geometría es una transformación que conserva las líneas rectas y las relaciones de distancias entre puntos que están en una línea recta. No preserva necesariamente ángulos o longitudes, pero tiene la propiedad de que conjuntos de líneas paralelas seguirá siendo paralelo el uno al otro después de una transformación afín.

Las transformaciones afines consisten en una transformación lineal seguida de una translación y preserva las siguientes características:

- Los puntos que estaban sobre una misma línea antes de la transformación se encuentran en la misma línea tras aplicar la transformación.

- Se conservan las proporciones /relaciones de las distancias que hay entre tres puntos distintos P1,P2,P3, la distancia del punto P1 al punto P2 y la distancia del punto P2 al punto P3 son proporcionales al aplicar la transformación afín.

Una transformación afín tiene esta forma en geometría:

Y → Ax + b, Donde A es una matriz y b un vector.

Ya que en algebra vectorial se usa la multiplicación de matrices para representar transformaciones lineales y la suma de vectores para la translación y la formula anterior quedaría de la siguiente manera.

$$\begin{pmatrix} y1 \\ y2 \\ . \\ . \\ . \\ yn \\ \end{pmatrix} = \begin{pmatrix} a11 \ldots . a1n \\ : \quad :::: \quad : \\ an1 \ldots . ann \end{pmatrix} \begin{pmatrix} x1 \\ . \\ . \\ xn \end{pmatrix} + \begin{pmatrix} b1 \\ . \\ . \\ bn \end{pmatrix}$$

Las transformaciones afines que nos interesan en este proyecto (además de las translaciones), son las siguientes:

Rotación: Movimiento de cambio de orientación de un cuerpo o un sistema de referencia de forma que una línea (llamada eje de rotación) o un punto permanece fijo.

Homotecia: Es una transformación afín que, a partir de un punto fijo, multiplica todas las distancias por un mismo factor (que le llamamos estiramiento).

5.2.2 Complejidad de algoritmos.

Un algoritmo es un conjunto de operaciones sistemáticas ordenadas que nos permiten encontrar una solución a un problema.

Un algoritmo debe poseer las siguientes características:

- Precisión: Un algoritmo debe expresarse sin ambigüedad.
- Determinismo: Todo algoritmo debe responder del mismo modo antes las mismas condiciones.
- Finito: La descripción de un algoritmo debe ser finita.
- Eficiente: Un algoritmo es eficiente cuantos menos recursos en tiempo, espacio (de memoria) y procesadores que consume.

La teoría computacional es la rama que se centra en la clasificación de los problemas computacionales de acuerdo a su dificultad que se tiene para resolverlos, la teoría computacional trata de clasificar los problemas que puedan o no ser resueltos con una determinada cantidad de recursos específicos.

La forma para medir la complejidad de un algoritmo es mediante el número de operaciones con respecto del tamaño de la entrada “n”, y por lo general se utiliza la notación O, la cual describe el comportamiento de una función cuando esta tiende al límite, cuando el tamaño de la muestra es grande o tiende a un valor infinito.

Un algoritmo que hace 3n + 12 operaciones con una entrada de tamaño n, y otro algoritmo que hace 16n – 24 operaciones con el mismo tamaño de entra, pertenecen a la misma familia O(n).

Un algoritmo que hace n^2 + 13n - 4 operaciones con una entrada de tamaño n, y otro algoritmo que hace n^2 + 3 operaciones con el mismo tamaño de entra, pertenecen a la misma familia O(n^2).

FUNCIÓN	COMPLEJIDAD	CUANDO OCURRE
O(1)	Orden constante	se ejecutan una vez
O(log n)	Orden logarítmico	con iteración o recursión no estructural
O(n)	Orden lineal	bucles simples siempre que la complejidad de las instrucciones interiores sea constante.
O(n log n)	Orden cuasi-lineal	Se encuentra en algoritmos de tipo divide y vencerás
O(n^a)	Orden polinómico	*a* > 1. Si a crece, la complejidad del programa es bastante mala
O(2^2)	Orden exponencial	Se dan en subprogramas recursivos que contengan dos o más llamadas internas
O(n!)	Orden factorial	

Figura 7 Tabla de complejidad

Se identifica una Jerarquía de Ordenes de Complejidad que coincide con el orden de la tabla mostrada; jerarquía en el sentido de que cada orden de complejidad inferior tiene a las superiores como subconjuntos.

Capítulo 6 "Desarrollo del proyecto"

En esta sección se describirán las etapas y condiciones en las que se desarrolló este proyecto.

De aquí en adelante se considerara:

- $C = \{C_1, C_2, ..., C_k\}$ la configuración de puntos a buscar,
- y $S = \{S_1, S_2, \ldots, S_n\}$ como el espacio de búsqueda.

6.1 Precondiciones.

Para evitar problemas numéricos (redondeo, uso de fórmulas trigonométricas, etc.) se decidió restringir a los puntos de C y S a que tuvieran coordenadas enteras.

6.2 Diseño de algoritmos.

A continuación se da una explicación breve de que es lo que realizan cada uno de los algoritmos diseñados e implementados en este proyecto.

6.2.1 Algoritmo para buscar si hay una copia de la configuración sin modificaciones.

Intuitivamente el proceso efectuado por este algoritmo es el siguiente:

Se toman los puntos, C_1 y C_2, de la configuración a encontrar y se busca una pareja de puntos en el espacio, S_i y S_j, que estén a la misma distancia y con el mismo ángulo que C_1 y C_2. Si no encontramos ninguna pareja de puntos en el espacio que cumpla con dichas características entonces la configuración no está presente en el espacio. Si sí existen S_i y S_j procedemos a seleccionar el tercer punto C_3 de la configuración y se examinan sus características con respecto a C_1 y C_2 (distancias y ángulos). A continuación se busca un tercer punto S_t en el espacio que tenga las mismas características que C_3 pero con respecto a S_i y S_j. Y se prosigue buscando puntos en el espacio que correspondan con los puntos C_4 hasta C_k de la configuración, en cuanto a distancias y ángulos con respecto a C_1 y C_2. Si alguna etapa de este proceso falla se buscan otros dos puntos S'_i y S'_j adecuados y se repite el proceso.

Pseudo código:

1 Se seleccionan los puntos C_1 y C_2 de la configuración

2. Se escoge el punto S_i para que pueda ser C_1 de entre todos los puntos $\{S_1,S_2,\ldots\ldots,S_n\}$ {

3. Se escoge el punto S_j para que pueda ser C_2 de entre todos los puntos $\{S_1,..,S_n\}/\{S_i\}$ {

4. Calculamos la distancia y ángulo entre C_1 y C_2

5. Calculamos la distancia y ángulo entre S_i y S_j

6. Si las distancias y ángulos son iguales {

7. Se recorren los puntos de C desde w=3 hasta C_k {

8. Se escoge S_t para que sea C_w de entre todos los $\{S_1,S_2,.....,S_n\}/\{S_i,S_j\}$ {

9. Calculamos la distancia y ángulo entre C_w y C_1

10. Calculamos la distancia y ángulo entre C_w y C_2

11. Calculamos la distancia y ángulo entre S_k y S_i

12. Calculamos la distancia y ángulo entre S_k y S_j

13. Si las distancias y ángulos son iguales {

14. El punto S_t pertenece a la configuración

15. }

16. }

17. }

18. }

19. }

20. }

6.2.2 Algoritmo para buscar si hay una copia de la configuración con cambios de escala.

El proceso que hace este algoritmo es similar al anterior con unas pequeñas modificaciones, las cuales son:

- En la línea 4 y 5 no calculamos la distancia
- En la línea 6 no se verifica que las distancias sean iguales.
- Entre la línea 6 y 7 se pondría una instrucción para calcular el factor de estiramiento.
- En la línea 10,11,12,13 la distancia se multiplica por el factor de estiramiento.

6.2.3 Algoritmo para buscar si hay una copia de la configuración con rotación.

El proceso que hace este algoritmo es similar al primero con unas pequeñas modificaciones, las cuales son:

- En la línea 4 y 5 no calculamos el ángulo
- En la línea 6 no se verifica que los ángulos sean iguales.
- Entre la línea 6 y 7 se pondría una instrucción para calcular el ángulo de rotación.
- En la línea 10,11,12,13 el ángulo se multiplica por el factor de rotación.

6.2.4 Algoritmo para buscar si hay una copia de la configuración con rotación y cambios de escala.

El proceso que hace este algoritmo es similar al primero con unas pequeñas modificaciones, las cuales son:

- No se realizan las líneas 4, 5 y 6.
- Entre la línea 6 y 7 se pondría una instrucción para calcular el ángulo de rotación y el factor de estiramiento.
- En la línea 10,11,12,13 el ángulo se multiplica por el factor de rotación y la distancia se multiplica por el factor de estiramiento.

6.3 Implementación.

6.3.1 Lenguaje de programación.

El lenguaje de programación que se utilizó para implementar los algoritmos anteriormente mencionados fue C++, se escogió porque en la etapa de diseño de los algoritmos notamos que se simplificaría el desarrollo si se utilizaba el paradigma de programación orientado a objetos y se C++ cuenta con él, además de que es un lenguaje de programación muy conocido y es posible de

instalarlo en varios de los sistemas operativos que hay en el mercado como lo es en el caso de Windows, Linux, Mac os, etc.

Otras de las cuestiones que nos llevó a decidir programar en C++ fue la velocidad de ejecución, si solo nos interesara eso C sería el mejor lenguaje de programación para hacer eso pero uno de sus inconvenientes es que no cuenta con el paradigma de programación orientada a objetos.

Se descartaron otros lenguajes de programación como lo fue C# y java por lo siguiente:

- C# debido a que es muy lento en cuestión de velocidad de ejecución de los algoritmos aunque este lenguaje de programación cuenta con el paradigma de la programación orientada a objetos.

- Java debido a que tiene que crear una máquina virtual para poder ejecutar los algoritmos y eso es muy lento en cuestión de ejecución y lo que nosotros requerimos es velocidad aunque este lenguaje de programación cuenta con el paradigma de la programación orientada a objetos.

6.3.2 Decisiones de diseño.

Las condiciones que tuvimos en cuenta para el diseño de la aplicación son las siguientes:

- No interfaz gráfica.
- Línea de comandos.

No cuenta con una interfaz gráfica porque la planteamos para que fuera una herramienta para que otros programas más extensos lo puedan utilizar como un módulo.

Se ejecuta desde la línea de comandos porque lo que nos interesa es que se ejecute rápidamente y los resultados que nos arrojan, y no la programamos de forma interactiva porque es lenta.

6.3.3 Clases utilizadas.

Estas son las clases que se elaboraron para la implementación de los algoritmos:

- Clase punto

  Es la clase básica de nuestro programa, debido a que trabajamos con puntos y estos tienen coordenadas enteras y está definida de la siguiente forma:

Punto
+ coord_x: int + coord_y: int
+ punto(int x=0, int y=0) + void imprime() + int get_coord_x() + int get_coord_y()

Figura 8 Clase punto

La clase contiene:

- Dos atributos de tipo entero, que son las coordenadas (x,y) de los puntos.
- Tiene una función constructora.
- La función imprime como su nombre lo dice imprime las coordenadas (x,y) en pantalla del punto.
- Las funciones get_coord_x y get_coord_y obtienen el valor de las coordenadas del punto.

- Clase pareja

  Esta clase es la que se encarga de hacer los cálculos matemáticas de propiedades de parejas de puntos, está definida de la siguiente manera:

Pareja
+ inicio: punto + final: punto
+ pareja (inicio: punto, final: punto) + int distancia () + double angulo()

Figura 9 Clase pareja

Que contiene la clase pareja:

- Dos atributos punto a los cuales llamamos inicio y final y sobre estos trabajaremos.
- Cuenta con una constructora.
- La función int distancia regresa el valor de la distancia entre el punto inicial y el punto final, pero no con raíz cuadrada para evitar problemas de redondeo.
- La función angulo regresa el valor del ángulo que forma el segmento de recta de los puntos iniciales y finales con respecto al eje x.

- Clase cunas

Esta clase tiene como función principal decidir si tres puntos del espacio cumplen con las mismas características que otros tres puntos de la configuración.

Está definida de la siguiente manera:

Cunas
+ base_fijo1: punto + base_fijo2: punto + base_movil: punto + explorar_fijo1: punto + explorar_fijo2: punto + explorar_movil: punto
+ cunas(base_fijo1:punto,base_fijo2: punto) + void asigna_explorar_fijos(explorar_fijo1: punto, explorar_fijo2: punto) + void asigna_base_movil(base_movil: punto) + void asigna_explorar_movil(explorar_movil: punto) + int cumple(factor: int) + int vuelta_base() +int vuelta_explorar()

Figura 10 clase cunas

El nombre de la clase era cuñas pero como no se puede poner la “ñ” para programar se puso cunas, el nombre está dado debido a que los segmentos de recta que unen los tres puntos parecen una cuña como se ve en la figura

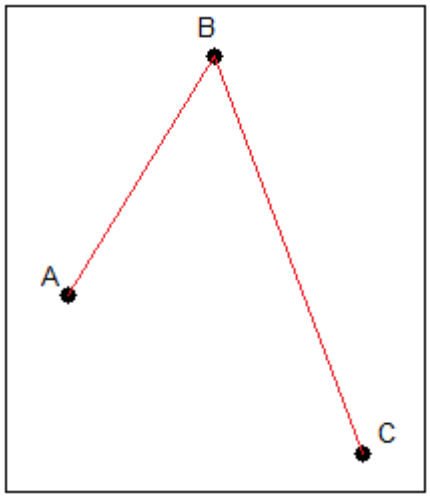


Figura 11 cunas

Lo que contiene la clase es lo siguiente:

- Tres objetos base de tipo punto que son los puntos de la configuración.

- Tres objetos explorar de tipo punto que son los puntos del espacio.
- Cuenta con una constructora.
- La función asigna_explorar_fijos asigna a los dos puntos del espacio que son candidatos a pertenecer a la configuración.
- La función asigna_base_movil asigna el siguiente punto de la configuración.
- La función asigna_explorar_movil asigna el siguiente punto del espacio que es candidato a pertenecer a la configuración.
- La función int cumple regresa un valor positivo si los tres puntos del espacio tienen las mismas características que los tres puntos de la configuración.
- La función int vuelta_base regresa los valores siguientes: si fue (-) la vuelta es a la derecha, si fue (+) la vuelta es a la izquierda y si fue (0) no hubo vuelta.
- La función int vuelta_explorar regresa los valores siguientes: si fue (-) la vuelta es a la derecha, si fue (+) la vuelta es a la izquierda y si fue (0) no hubo vuelta.

6.4 Dificultades y como se resolvieron.

A continuación se enlistan algunas de las dificultades que se tuvieron a lo largo del proyecto y la forma en que se resolvieron:

- En nuestros algoritmos se hacen muchos cálculos aritméticos, los cuales pueden llevarnos a problemas de precisión esta dificultad se resolvió con coordenadas enteras no sacando la raíz cuadrada a las distancias y no transformando los ángulos de radianes a grados.

- Dado dos puntos S_i y S_j se quiere encontrar un punto S_t que este a una distancia de S_i y S_j, lamentablemente hay dos puntos que son candidatos a ser S_t, el problema que tuvimos fue saber qué punto era, y sacar el ángulo lleva a problemas de cálculo y se resolviendo utilizando el producto cruz y esto se observa en las figuras .

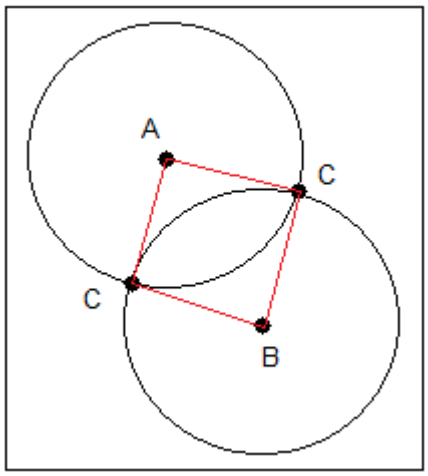


Figura 12 puntos alternativos

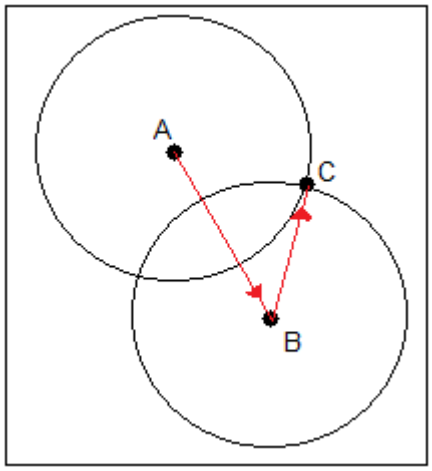


Figura 13 camino de un punto a otro

6.5 Tiempo de desarrollo.

La realización del proyecto fue de la siguiente manera:

- Las dos primeras semanas se estudiaron libros de análisis y diseño de algoritmos y se empezó a diseñar los algoritmos.
- Las siguientes siete semanas se terminó de diseñar los algoritmos, se implementaron y se realizaron las pruebas.

Las dos semanas restantes se elaboró el reporte escrito del proyecto.

Capítulo 7 “Resultados”

Lo que se realizó en este proyecto son 4 aplicaciones, las cuales resuelven cada uno de los objetivos específicos, mencionados en el capítulo 4.

Las 4 aplicaciones se ejecutan mediante la línea de comandos del Sistema Operativo, estas reciben 3 parámetros exactamente:

1. Nombre de la aplicación que se ejecuta.
2. Archivo de texto que contiene la configuración que buscamos.
3. Archivo de texto que contiene el espacio de búsqueda.

Entonces cada aplicación se correría de la siguiente manera:

$> [nombre de la aplicación] [archivo de texto configuración] [archivo de texto espacio]

Los dos archivos de texto que se esperan en la aplicación tiene el siguiente formato: cada línea contiene las coordenadas de un punto, dos números enteros separados por una coma ”,” ,sino esta en este formato la aplicación no puede procesar la información.

Si no se introduce en esta forma las cosas, la aplicación cuenta con una pequeña ayuda que nos indica de una forma clara como es que se tiene que ejecutar la aplicación.

```
dani@dani-Aspire-one:~/Escritorio/CD$ ./busca_normal


uso: busca_normal <archivo1.txt> <archivo2.txt>

busca una configuracion de puntos a dentro de un espacio de puntos


            El archivo1 debe contener la configuracion que buscamos
            El archivo2 debe contener el espacio de busqueda
            Los dos archivos los esperamos con n numero de lineas, con la forma
'entero,entero'
```

Figura 14 ayuda de la aplicación

Cuando la aplicación fue ejecutada correctamente se muestra en pantalla alguno de los resultados siguientes:

Si encontró una copia de la configuración que buscamos, nos muestra los puntos que la conforman.

```
dani@dani-Aspire-one:~/Escritorio/CD$ ./busca_normal /home/dani/Escritorio/confi
guracion1.txt /home/dani/Escritorio/espacio1.txt
Aviso: Una linea de tu archivo /home/dani/Escritorio/configuracion1.txt no cumpl
e con el formato esperado.



Estos son los puntos a buscar:
5,8   9,3   4,1   3,4   1,2   7,6

El espacio en donde buscamos tiene este numero de puntos:400


BUSCA SI ESTA LA CONFIGURACION ORIGINAL
Si se encuentra la configuracion y estos son su puntos:
5,8   9,3   4,1   3,4   2,7   7,6
```

Figura 15 si se encontró la configuración

Si no encontró una copia de la configuración que buscamos, despliega un letrero diciendo que no se encontró.

```
dani@dani-Aspire-one:~/Escritorio/CD$ ./busca_normal /home/dani/Escritorio/confi
guracion1.txt /home/dani/Escritorio/espacio4.txt
Aviso: Una linea de tu archivo /home/dani/Escritorio/configuracion1.txt no cumpl
e con el formato esperado.



Estos son los puntos a buscar:
5,8   9,3   4,1   3,4   1,2   7,6

El espacio en donde buscamos tiene este numero de puntos:400


BUSCA SI ESTA LA CONFIGURACION ORIGINAL
No se encuentra la configuracion
```

Figura 16 No se encontró la configuración

Una vez compilado el proyecto los nombres que reciben nuestras aplicaciones son:

- Busca_normal.
- Busca_aumento.
- Busca_rotada.
- Busca_aumento_rotada.

Capítulo 8 "Análisis y discusión de los resultados"

En cada una de las siguientes secciones se analizara una característica de nuestros algoritmos y programas.

8.1 Complejidad de los algoritmos.

Al medir la complejidad de un algoritmo siempre interesa el "peor caso" y es lo que se estudia en esta sección.

8.1.1 Buscando si hay una copia de la configuración sin cambios.

Para determinar si existe una copia de la configuración en el espacio, en el peor de los casos lo que realiza nuestro algoritmo es lo siguiente:

- Escoger la pareja de puntos de la configuración que buscamos, C_1 y C_2, se realiza con complejidad O(1).
- Buscar todas las posibles parejas S_i y S_j en el espacio, que correspondan con los puntos C_1 y C_2, se realiza con complejidad $O(n^2)$ porque se cuenta con "n" opciones para seleccionar a S_i y esas mismas "n" opciones para S_j.
- Buscar cada uno de los k-2 puntos restantes de la configuración que se busca, si existe alguno de los (n-2) puntos que no son seleccionados en el espacio se realiza con complejidad $O((k-2)*(n-2)) = O(Kn)$.

Juntando todas las complejidades que se obtuvieron, resulta que el algoritmo es de complejidad $O((n^2) * (kn)) = O(kn^3)$.

8.1.2 Complejidad de los otros tres algoritmos

En el capítulo 6 se menciona que los tres algoritmos restantes son una modificación de este algoritmo principal, las modificaciones que se realizan en cada uno de los algoritmos no afectan la complejidad de estos asi que todos los algoritmos tienen complejidad $O(kn^3)$.

8.2 Ejemplo de ejecución.

Se mostrara el resultado que se obtiene en nuestros programas al realizar la búsqueda de la siguiente configuración de puntos con coordenadas: (5,8),(9,3),(4,1),(3,4),(1,2),(7,6). En los siguientes espacios de búsqueda.

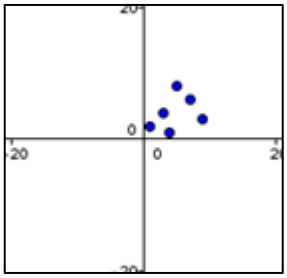


Figura 17 Configuración de 6 puntos

8.2.1 Buscando la configuración sin cambios.

Buscando la configuración de la figura 17, en el siguiente espacio de búsqueda con estas coordenadas:
(-7,19),(-14,-48),(21,-24),(30,10),(14,16),(-43,-3),(33,-21),(13,43),(47,-6),(-21,-12),(43,-44),(-46,5),(44,34),(-27,-32),(-30,47),(-1,-22),(23,-10),(21,-36),(19,51),(-13,46),(-45,-37),(-26,-15),(25,16),(-7,-37),(5,20),(-1,-4),(14,9),(-11,11),(-25,-7),(-19,30),(-32,-13),(42,-6),(40,-42),(-8,-6),(16,0),(-2,-43),(42,-19),(22,2),(-42,-47),(45,0),(-19,-25),(36,6),(8,-8),(18,28),(-17,-40),(-4,-9),(-22,-25),(-11,-10),(-30,34),(-19,-7),(-15,-33),(-9,10),(-44,-18),(29,-42),(25,38),(-27,-3),(-24,24),(22,-19),(29,25),(49,-36),(38,42),(13,-12),(7,19),(7,26),
(9,8),(13,3),(8,1),(7,4),(5,2),(11,6),(-17,4),(2,2),(-7,-24),(18,-18),(-41,43),(-41,-11),(9,39),(5,35),(-3,-39),(-39,10),(-27,40),(-26,-2),(-42,-18),(-35,20),(-48,43),(14,7),(-38,11),(-24,-11),(0,35),(47,-7),(-46,2),(43,-12),(26,-28),(48,-27),(0,51),(20,36),(33,-14),(5,51),(-30,-10),(-48,40),(-21,19),(-20,45),(0,35),(-41,-27),(-38,-31),(-35,-34),(-39,-32),(-13,3),(-48,1),(-29,8),(50,-45),(-24,-40),(-4,-39),(41,-46),(47,37),(45,-5),(-25,39),(-34,-45),(0,-48),(10,-30),(32,48),(50,33),(41,50),(-39,9),(24,-26),(-10,44),(-10,31),(42,9),(10,43),(-33,40),(8,-37),(-46,-14),(24,7),(-20,-2),(14,38),(27,-15),(21,-6),(-4,-32),(33,50),(-26,3),(-27,51),(9,28),(44,41),(27,-36),(-48,-38),(-45,21),(13,40),(-47,41),(7,-25),(-46,37),(34,37),(40,-22),(-31,9),(-16,-16),(21,6),(-27,41),(28,-19),(20,44),(-23,7),(-14,1),(-7,-36),(-3,12),(-30,5),(-9,-25),(31,48),(39,-19),(1,-11),(18,1),(45,47),(49,-32),(38,-43),(40,34),(7,-14),(-34,-47),(-32,23),(38,15),(-35,7),(37,5),(-36,-40),(-16,-3),(-35,8),(-27,10),(-2,34),(33,-4),(48,-26),(-19,13),(-13,2),(25,18),(-4,11),(44,-9),(5,-24),(6,-38),(-3,1),(38,-35),(26,-26),(20,-30),(39,-43),(10,43),(-46,-23),(29,-34),(-34,-24),(-14,26),(24,11),(-15,22),(39,49),(-30,29),(25,22),(15,20),(44,37),(-46,32),(-35,-21),(-46,51),(-21,-23),(-5,-24),(-25,24),(13,33),(-45,-16),(-43,45),(-23,-17),(44,-6),(-26,38),(16,-48),(39,12),(-35,26),(22,22),(-13,-15),(-37,12),(48,19),(37,2),(-8,46),(47,-24),(-29,-23),(28,46),(10,-46),(23,18),(30,45),(3,36),(-30,16),(-29,4),(-48,39),(12,-22),(-38,9),(22,-33),(28,-21),(-5,10),(16,-39),(34,38),(17,39),(29,26),(-23,-21),(-19,-20),(-25,-28),(-46,14),(-25,48),(-11,13),(47,-23),(16,12),(-46,-32),(-18,-22),(-37,23),(-37,-1),(5,-28),(42,-24),(40,15),(-8,3),(14,-19),(-48,-35),(10,30),(17,-41),(29,-48),(10,-9),(-45,12),(9,-24),(29,-40),(-35,39),(-47,2),(12,-20),(45,36),(-43,-8),(-37,-44),(-13,8),(24,2),(-25,37),(8,-32),(-22,9),(-22,9),(-11,23),(21,13),(48,-26),(-31,-36),(-31,48),(37,-7),(-25,-19),(-19,17),(11,-16),(48,7),(5,14),(36,-14),(6,24),(9,21),(-16,15),(-41,35),(-37,-13),(19,0),(27,-10),(-25,-6),(6,-37),(-7,27),(11,-23),(-27,22),(-22,-14),(-43,35),(2,50),(31,-47),(45,-14),(-11,-14),(8,45),(28,-43),(14,0),(33,-48),(-35,-7),(7,7),(-6,14),(-

37,29),(-24,30),(4,-5),(48,25),(-8,-35),(27,24),(-30,-38),(-31,-16),(-36,47),(21,-17),(-8,40),(37,42),(49,41),(42,-3),(5,-34),(3,-8),(-4,10),(-13,11),(44,-43),(16,33),(-45,-19),(27,-40),(44,49),(1,8),(13,-21),(19,-7),(-19,-8),(-35,26),(-47,29),(-33,35),(-35,44),(-24,-47),(44,11),(22,-20),(-21,36),(27,38),(50,22),(39,-1),(-44,-45),(-27,15),(-42,15),(-38,23),(41,-8),(16,-6),(-29,-35),(43,-44),(-30,-16),(2,-43),(27,-9),(-45,-26),(50,-1),(36,0),(23,16),(-35,27),(-3,-36),(-2,30),(21,14),(-29,37),(41,-4),(17,-8),(-3,-40),(-30,22),(-47,-25),(-16,24),(4,39),(22,15),(-47,-45),(-25,-21),(-48,21),(-33,17),(-20,-5),(-1,40),(-5,-11),(-39,15),(1,33),(40,-6),(-40,12).

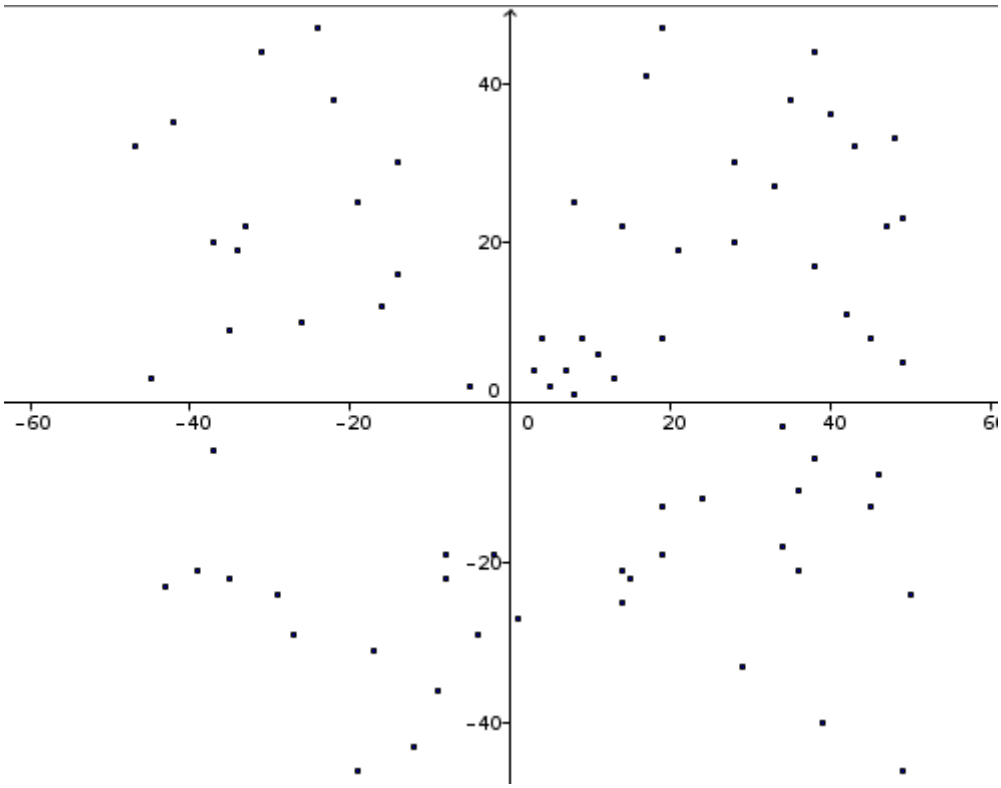


Figura 18 Espacio de búsqueda

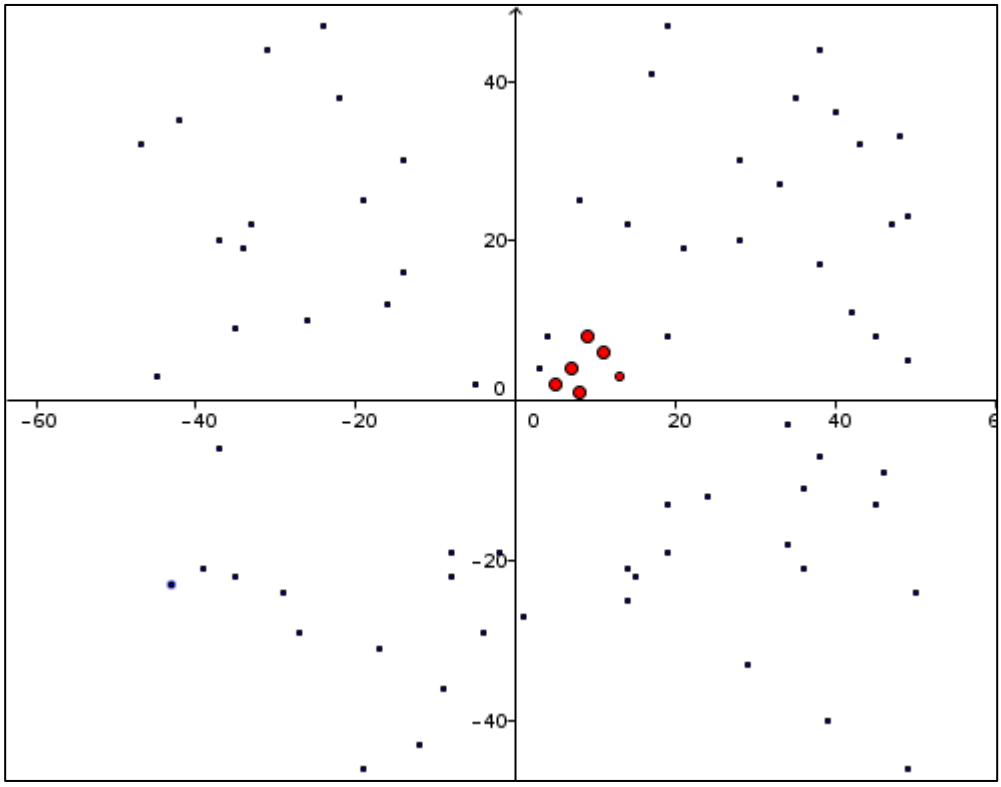


Figura 19 Configuracion encontrada

Si se encontró una copia de la configuración en las siguientes coordenadas: (9,8),(13,3),(8,1),(7,4),(5,2),(11,6)

8.2.2 Buscando si hay una copia de la configuración con cambios de escala.

Buscando la configuración de la figura 17, en el siguiente espacio de búsqueda con estas coordenadas:
(-7,19),(-14,-48),(21,-24),(30,10),(14,16),(-43,-3),(33,-21),(13,43),(47,-6),(-21,-12),(43,-44),(-46,5),(44,34),(-27,-32),(-30,47),(-1,-22),(23,-10),(21,-36),(19,51),(-13,46),(-45,-37),(-26,-15),(25,16),(-7,-37),(5,20),(-1,-4),(14,9),(-11,11),(-25,-7),(-19,30),(-32,-13),(42,-6),(40,-42),(-8,-6),(16,0),(-2,-43),(42,-19),(22,2),(-42,-47),(45,0),(-19,-25),(36,6),(8,-8),(18,28),(-17,-40),(-4,-9),(-22,-25),(-11,-10),(-30,34),(-19,-7),(-15,-33),(-9,10),(-44,-18),(29,-42),(25,38),(-27,-3),(-24,24),(22,-19),(29,25),(49,-36),(38,42),(13,-12),(7,19),(7,26),(-17,4),(2,2),(-7,-24),(18,-18),(-41,43),(-41,-11),(9,39),(5,35),(-3,-39),(-39,10),(-27,40),(-26,-2),(-42,-18),(-35,20),(-48,43),(14,7),(-38,11),(-24,-11),(0,35),(47,-7),(-46,2),(43,-12),(26,-28),(48,-27),(0,51),(20,36),(33,-14),(5,51),(-30,-10),(-48,40),(-21,19),(-20,45),(0,35),(-41,-27),(-38,-31),(-35,-34),(-39,-32),(-13,3),(-48,1),(-29,8),(50,-45),(-24,-40),(-4,-39),(41,-46),(47,37),(45,-5),(-25,39),(-34,-45),(0,-48),(10,-30),(32,48),(50,33),(41,50),(-39,9),(24,-26),(-10,44),(-10,31),(42,9),(10,43),(-33,40),(8,-37),(-46,-14),(24,7),(-20,-2),(14,38),(27,-15),(21,-6),(-4,-32),(33,50),(-26,3),(-27,51),(9,28),(44,41),(27,-36),(-48,-38),(-45,21),(13,40),(-47,41),(7,-25),(-46,37),(34,37),(40,-22),(-31,9),(-16,-16),(21,6),(-27,41),(28,-19),(20,44),(-23,7),(-14,1),(-7,-36),(-3,12),(-30,5),(-9,-

25),(31,48),(39,-19),(1,-11),(18,1),(45,47),(49,-32),(38,-43),(40,34),(7,-14),(-34,-47),(-32,23),(38,15),(-35,7),(37,5),(-36,-40),(-16,-3),(-35,8),(-27,10),(-2,34),(33,-4),(48,-26),(-19,13),(-13,2),(25,18),(-4,11),(44,-9),(5,-24),(6,-38),(-3,1),(38,-35),(26,-26),(20,-30),(39,-43),(10,43),(-46,-23),(29,-34),(-34,-24),(-14,26),(24,11),(-15,22),(39,49),(-30,29),(25,22),(15,20),(44,37),(-46,32),(-35,-21),(-46,51),(-21,-23),(-5,-24),(-25,24),(13,33),(-45,-16),(-43,45),(-23,-17),(44,-6),(-26,38),(16,-48),(39,12),(-35,26),(22,22),(-13,-15),(-37,12),(48,19),(37,2),(-8,46),(47,-24),(-29,-23),(28,46),(10,-46),(23,18),(30,45),(3,36),(-30,16),(-29,4),(-48,39),(12,-22),(-38,9),(22,-33),(28,-21),(-5,10),
(15,24),(27,9),(12,3),(9,12),(3,6),(21,18),(16,-39),(34,38),(17,39),(29,26),(-23,-21),(-19,-20),(-25,-28),(-46,14),(-25,48),(-11,13),(47,-23),(16,12),(-46,-32),(-18,-22),(-37,23),(-37,-1),(5,-28),(42,-24),(40,15),(-8,3),(14,-19),(-48,-35),(10,30),(17,-41),(29,-48),(10,-9),(-45,12),(9,-24),(29,-40),(-35,39),(-47,2),(12,-20),(45,36),(-43,-8),(-37,-44),(-13,8),(24,2),(-25,37),(8,-32),(-22,9),(-22,9),(-11,23),(21,13),(48,-26),(-31,-36),(-31,48),(37,-7),(-25,-19),(-19,17),(11,-16),(48,7),(5,14),(36,-14),(6,24),(9,21),(-16,15),(-41,35),(-37,-13),(19,0),(27,-10),(-25,-6),(6,-37),(-7,27),(11,-23),(-27,22),(-22,-14),(-43,35),(2,50),(31,-47),(45,-14),(-11,-14),(8,45),(28,-43),(14,0),(33,-48),(-35,-7),(7,7),(-6,14),(-37,29),(-24,30),(4,-5),(48,25),(-8,-35),(27,24),(-30,-38),(-31,-16),(-36,47),(21,-17),(-8,40),(37,42),(49,41),(42,-3),(5,-34),(3,-8),(-4,10),(-13,11),(44,-43),(16,33),(-45,-19),(27,-40),(44,49),(1,8),(13,-21),(19,-7),(-19,-8),(-35,26),(-47,29),(-33,35),(-35,44),(-24,-47),(44,11),(22,-20),(-21,36),(27,38),(50,22),(39,-1),(-44,-45),(-27,15),(-42,15),(-38,23),(41,-8),(16,-6),(-29,-35),(43,-44),(-30,-16),(2,-43),(27,-9),(-45,-26),(50,-1),(36,0),(23,16),(-35,27),(-3,-36),(-2,30),(21,14),(-29,37),(41,-4),(17,-8),(-3,-40),(-30,22),(-47,-25),(-16,24),(4,39),(22,15),(-47,-45),(-25,-21),(-48,21),(-33,17),(-20,-5),(-1,40),(-5,-11),(-39,15),(1,33),(40,-6),(-40,12).

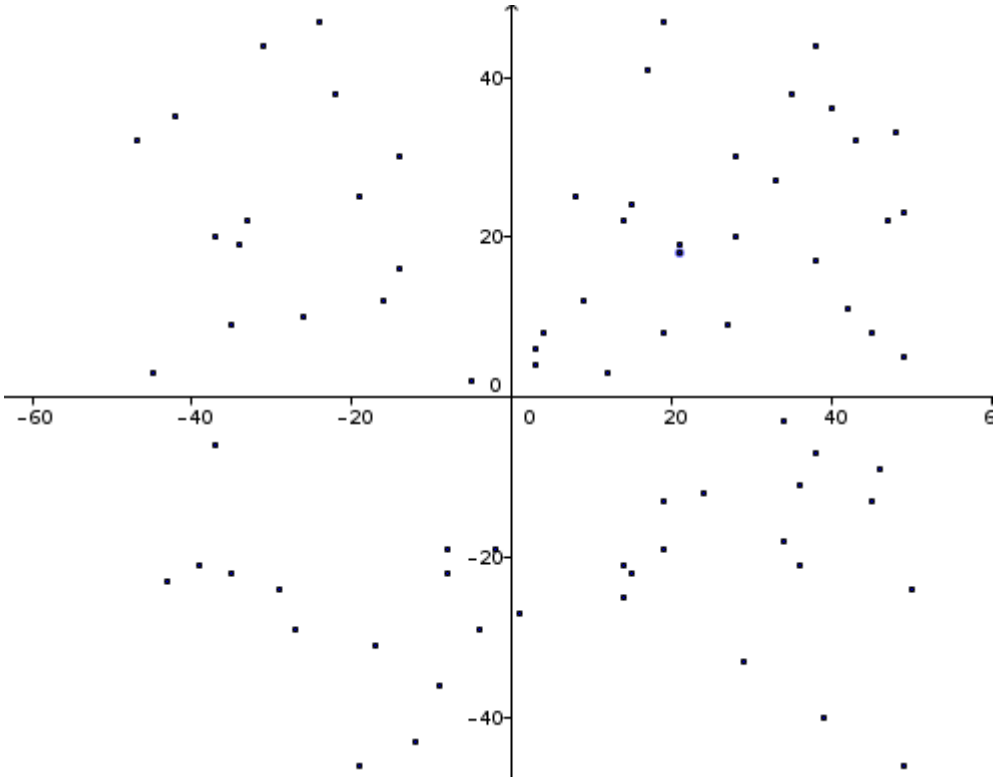


Figura 20 Espacio de búsqueda

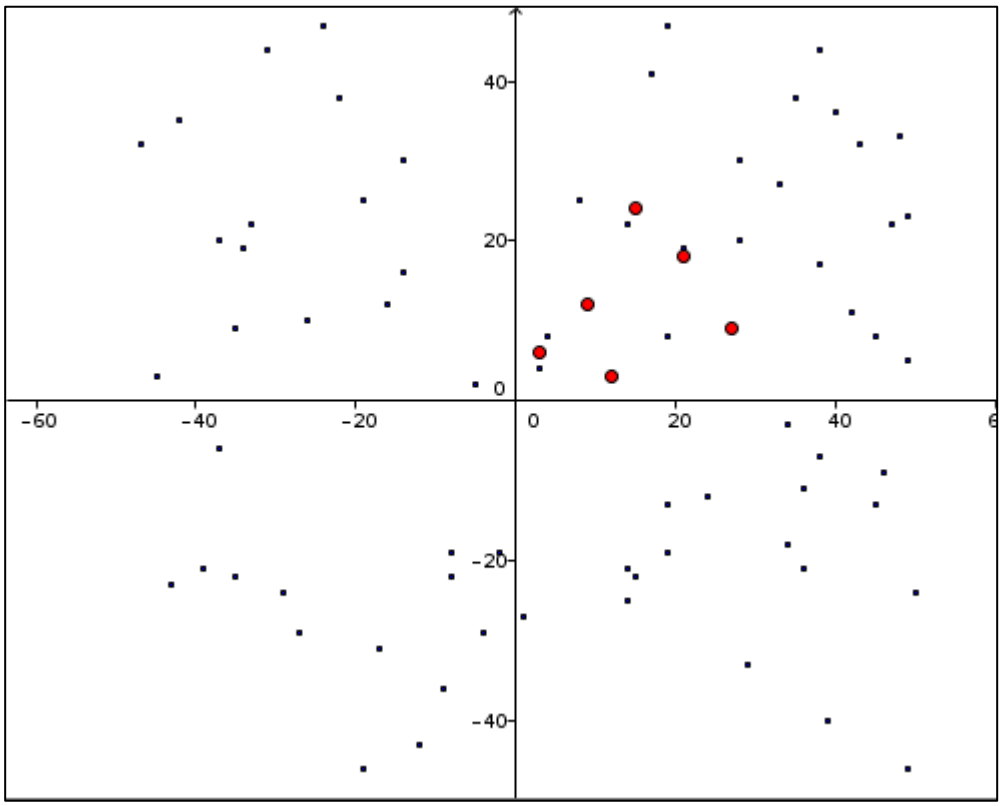


Figura 21 Configuración encontrada

La configuración que se encontró es, con las siguientes coordenadas:

(15,24),(27,9),(12,3),(9,12),(3,6),(21,18).

8.2.3 Buscando si hay una copia de la configuración con cambios de rotación.

Buscando la configuración de la figura 17, en el siguiente espacio de búsqueda con estas coordenadas:
(-7,19),(-14,-48),(21,-24),(30,10),(14,16),(-43,-3),(33,-21),(13,43),(47,-6),(-21,-12),(43,-44),(-46,5),(44,34),(-27,-32),(-30,47),(-1,-22),(23,-10),(21,-36),(19,51),(-13,46),(-45,-37),(-26,-15),(25,16),(-7,-37),(5,20),(-1,-4),(14,9),(-11,11),(-25,-7),(-19,30),(-32,-13),(42,-6),(40,-42),(-8,-6),(16,0),(-2,-43),(42,-19),(22,2),(-42,-47),(45,0),(-19,-25),(36,6),(8,-8),(18,28),(-17,-40),(-4,-9),(-22,-25),(-11,-10),(-30,34),(-19,-7),(-15,-33),(-9,10),(-44,-18),(29,-42),(25,38),(-27,-3),(-24,24),(22,-19),(29,25),(49,-36),(38,42),(13,-12),(7,19),(7,26),(-17,4),(2,2),(-7,-24),(18,-18),(-41,43),(-41,-11),(9,39),(5,35),(-3,-39),(-39,10),(-27,40),(-26,-2),(-42,-18),(-35,20),(-48,43),(14,7),(-38,11),(-24,-11),(0,35),(47,-7),(-46,2),(43,-12),(26,-28),(48,-27),(0,51),(20,36),(33,-14),(5,51),(-30,-10),(-48,40),(-21,19),(-20,45),(0,35),(-41,-27),(-38,-31),(-35,-34),(-39,-32),(-13,3),(-48,1),(-29,8),(50,-45),(-24,-40),(-4,-39),(41,-46),(47,37),(45,-5),(-25,39),(-34,-45),(0,-48),(10,-30),(32,48),(50,33),(41,50),(-39,9),(24,-26),(-10,44),(-10,31),(42,9),(10,43),(-33,40),(8,-37),(-46,-14),(24,7),(-20,-2),(14,38),(27,-15),(21,-6),(-4,-32),(33,50),(-26,3),(-27,51),(9,28),(44,41),(27,-36),(-48,-38),(-45,21),(13,40),(-47,41),(7,-25),(-46,37),(34,37),(-8,-9),(-3,-13),(-1,-8),(-4,-7),(-2,-5),(-6,-11),(40,-22),(-31,9),(-16,-16),(21,6),(-27,41),(28,-19),(20,44),(-23,7),(-14,1),(-7,-36),(-3,12),(-30,5),(-9,-25),(31,48),(39,-19),(1,-11),(18,1),(45,47),(49,-32),(38,-43),(40,34),(7,-14),(-34,-47),(-32,23),(38,15),(-35,7),(37,5),(-36,-40),(-16,-3),(-35,8),(-27,10),(-2,34),(33,-4),(48,-26),(-19,13),(-13,2),(25,18),(-4,11),(44,-9),(5,-24),(6,-38),(-3,1),(38,-35),(26,-26),(20,-30),(39,-43),(10,43),(-46,-23),(29,-34),(-34,-24),(-14,26),(24,11),(-15,22),(39,49),(-30,29),(25,22),(15,20),(44,37),(-46,32),(-35,-21),(-46,51),(-21,-23),(-5,-24),(-25,24),(13,33),(-45,-16),(-43,45),(-23,-17),(44,-6),(-26,38),(16,-48),(39,12),(-35,26),(22,22),(-13,-15),(-37,12),(48,19),(37,2),(-8,46),(47,-24),(-29,-23),(28,46),(10,-46),(23,18),(30,45),(3,36),(-30,16),(-29,4),(-48,39),(12,-22),(-38,9),(22,-33),(28,-21),(-5,10),(16,-39),(34,38),(17,39),(29,26),(-23,-21),(-19,-20),(-25,-28),(-46,14),(-25,48),(-11,13),(47,-23),(16,12),(-46,-32),(-18,-22),(-37,23),(-37,-1),(5,-28),(42,-24),(40,15),(-8,3),(14,-19),(-48,-35),(10,30),(17,-41),(29,-48),(10,-9),(-45,12),(9,-24),(29,-40),(-35,39),(-47,2),(12,-20),(45,36),(-43,-8),(-37,-44),(-13,8),(24,2),(-25,37),(8,-32),(-22,9),(-22,9),(-11,23),(21,13),(48,-26),(-31,-36),(-31,48),(37,-7),(-25,-19),(-19,17),(11,-16),(48,7),(5,14),(36,-14),(6,24),(9,21),(-16,15),(-41,35),(-37,-13),(19,0),(27,-10),(-25,-6),(6,-37),(-7,27),(11,-23),(-27,22),(-22,-14),(-43,35),(2,50),(31,-47),(45,-14),(-11,-14),(8,45),(28,-43),(14,0),(33,-48),(-35,-7),(7,7),(-6,14),(-37,29),(-24,30),(4,-5),(48,25),(-8,-35),(27,24),(-30,-38),(-31,-16),(-36,47),(21,-17),(-8,40),(37,42),(49,41),(42,-3),(5,-34),(3,-8),(-4,10),(-13,11),(44,-43),(16,33),(-45,-19),(27,-40),(44,49),(1,8),(13,-21),(19,-7),(-19,-8),(-35,26),(-47,29),(-33,35),(-35,44),(-24,-47),(44,11),(22,-20),(-21,36),(27,38),(50,22),(39,-1),(-44,-45),(-27,15),(-42,15),(-38,23),(41,-8),(16,-6),(-29,-35),(43,-44),(-30,-16),(2,-43),(27,-9),(-45,-26),(50,-1),(36,0),(23,16),(-35,27),(-3,-36),(-2,30),(21,14),(-29,37),(41,-4),(17,-8),(-3,-40),(-30,22),(-47,-25),(-

16,24),(4,39),(22,15),(-47,-45),(-25,-21),(-48,21),(-33,17),(-20,-5),(-1,40),(-5,-11),(-39,15),(1,33),(40,-6),(-40,12).

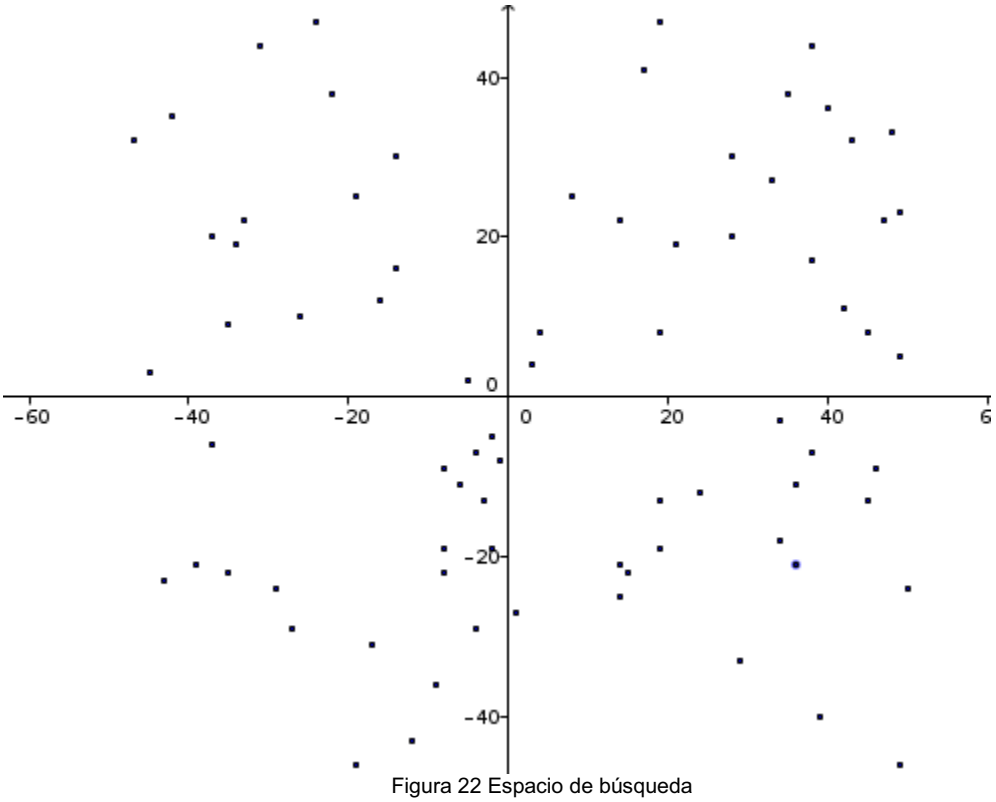


Figura 22 Espacio de búsqueda

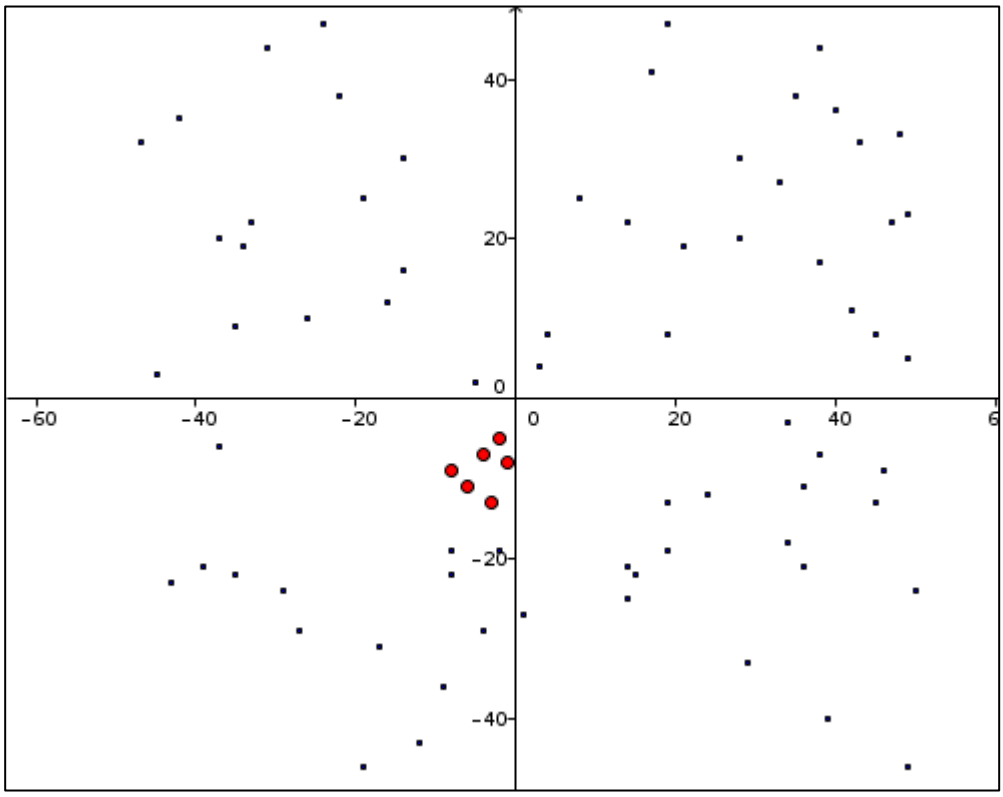


Figura 23 Configuración encontrada

La configuración que se encontró es, con las siguientes coordenadas:
(-8,-9),(-3,-13),(-1,-8),(-4,-7),(-2,-5),(-6,-11).

8.2.4 Buscando si hay una copia de la configuración con cambios de rotación y escala.

Buscando la configuración de la figura 17, en el siguiente espacio de búsqueda con estas coordenadas:
(-7,19),(-14,-48),(21,-24),(30,10),(14,16),(-43,-3),(33,-21),(13,43),(47,-6),(-21,-12),(43,-44),(-46,5),(44,34),(-27,-32),(-30,47),(-1,-22),(23,-10),(21,-36),(19,51),(-13,46),(-45,-37),(-26,-15),(25,16),(-7,-37),(5,20),(-1,-4),(14,9),(-11,11),(-25,-7),(-19,30),(-32,-13),(42,-6),(40,-42),(-8,-6),(16,0),(-2,-43),(42,-19),(22,2),(-42,-47),(45,0),(-19,-25),(36,6),(8,-8),(18,28),(-17,-40),(-4,-9),(-22,-25),(-11,-10),(-30,34),(-19,-7),(-15,-33),(-9,10),(-44,-18),(29,-42),(25,38),(-27,-3),(-24,24),(22,-19),(29,25),(49,-36),(38,42),(13,-12),(7,19),(7,26),(-17,4),(2,2),(-7,-24),(18,-18),(-41,43),(-41,-11),(9,39),(5,35),(-3,-39),(-39,10),(-27,40),(-26,-2),(-42,-18),(-35,20),(-48,43),(14,7),(-38,11),(-24,-11),(0,35),(47,-7),(-46,2),(43,-12),(26,-28),(48,-27),(0,51),(20,36),(33,-14),(5,51),(-30,-10),(-48,40),(-21,19),(-20,45),(0,35),(-41,-27),(-38,-31),(-35,-34),(-39,-32),(-13,3),(-48,1),(-29,8),(50,-45),(-24,-40),(-4,-39),(41,-46),(47,37),(45,-5),(-25,39),(-34,-45),(0,-48),(10,-30),(32,48),(50,33),(41,50),(-39,9),(24,-26),(-10,44),(-10,31),(42,9),(10,43),(-33,40),(8,-37),(-46,-14),(24,7),(-20,-2),(14,38),(27,-15),(21,-6),(-4,-32),(33,50),(-26,3),(-27,51),(9,28),(44,41),(27,-36),(-48,-38),(-45,21),(13,40),(-47,41),(7,-25),(-46,37),(34,37),(40,-22),(-31,9),(-16,-16),(21,6),(-27,41),(28,-19),(20,44),(-23,7),(-14,1),(-7,-36),(-3,12),(-30,5),(-9,-25),(31,48),(39,-19),(1,-11),(18,1),(45,47),(49,-32),(38,-43),(40,34),(7,-14),(-34,-47),(-32,23),(38,15),(-35,7),(37,5),(-36,-40),(-16,-3),(-35,8),(-27,10),(-2,34),(33,-4),(48,-26),(-19,13),(-13,2),(25,18),(-4,11),(44,-9),(5,-24),(-24,15),(-9,27),(-3,12),(-12,9),(-6,3),(-18,21),(6,-38),(-3,1),(38,-35),(26,-26),(20,-30),(39,-43),(10,43),(-46,-23),(29,-34),(-34,-24),(-14,26),(24,11),(-15,22),(39,49),(-30,29),(25,22),(15,20),(44,37),(-46,32),(-35,-21),(-46,51),(-21,-23),(-5,-24),(-25,24),(13,33),(-45,-16),(-43,45),(-23,-17),(44,-6),(-26,38),(16,-48),(39,12),(-35,26),(22,22),(-13,-15),(-37,12),(48,19),(37,2),(-8,46),(47,-24),(-29,-23),(28,46),(10,-46),(23,18),(30,45),(3,36),(-30,16),(-29,4),(-48,39),(12,-22),(-38,9),(22,-33),(28,-21),(-5,10),(16,-39),(34,38),(17,39),(29,26),(-23,-21),(-19,-20),(-25,-28),(-46,14),(-25,48),(-11,13),(47,-23),(16,12),(-46,-32),(-18,-22),(-37,23),(-37,-1),(5,-28),(42,-24),(40,15),(-8,3),(14,-19),(-48,-35),(10,30),(17,-41),(29,-48),(10,-9),(-45,12),(9,-24),(29,-40),(-35,39),(-47,2),(12,-20),(45,36),(-43,-8),(-37,-44),(-13,8),(24,2),(-25,37),(8,-32),(-22,9),(-22,9),(-11,23),(21,13),(48,-26),(-31,-36),(-31,48),(37,-7),(-25,-19),(-19,17),(11,-16),(48,7),(5,14),(36,-14),(6,24),(9,21),(-16,15),(-41,35),(-37,-13),(19,0),(27,-10),(-25,-6),(6,-37),(-7,27),(11,-23),(-27,22),(-22,-14),(-43,35),(2,50),(31,-47),(45,-14),(-11,-14),(8,45),(28,-43),(14,0),(33,-48),(-35,-7),(7,7),(-6,14),(-37,29),(-24,30),(4,-5),(48,25),(-8,-35),(27,24),(-30,-38),(-31,-16),(-36,47),(21,-17),(-8,40),(37,42),(49,41),(42,-3),(5,-34),(3,-8),(-4,10),(-13,11),(44,-43),(16,33),(-45,-19),(27,-40),(44,49),(1,8),(13,-21),(19,-7),(-19,-8),(-35,26),(-

47,29),(-33,35),(-35,44),(-24,-47),(44,11),(22,-20),(-21,36),(27,38),(50,22),(39,-1),(-44,-45),(-27,15),(-42,15),(-38,23),(41,-8),(16,-6),(-29,-35),(43,-44),(-30,-16),(2,-43),(27,-9),(-45,-26),(50,-1),(36,0),(23,16),(-35,27),(-3,-36),(-2,30),(21,14),(-29,37),(41,-4),(17,-8),(-3,-40),(-30,22),(-47,-25),(-16,24),(4,39),(22,15),(-47,-45),(-25,-21),(-48,21),(-33,17),(-20,-5),(-1,40),(-5,-11),(-39,15),(1,33),(40,-6),(-40,12).

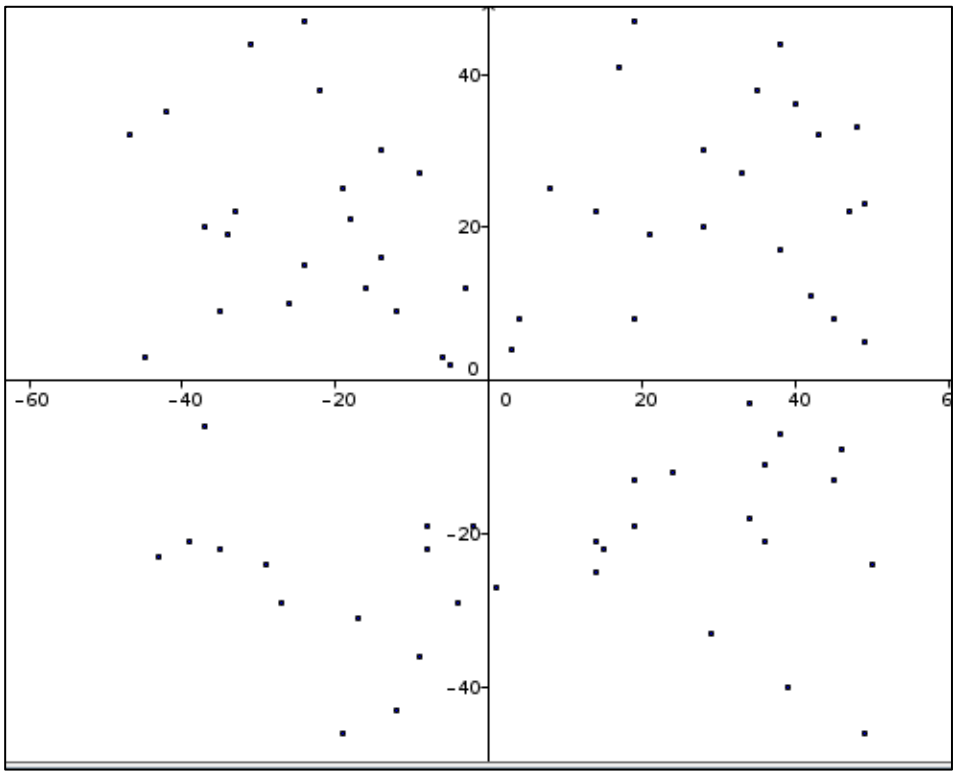


Figura 24 Espacio de búsqueda

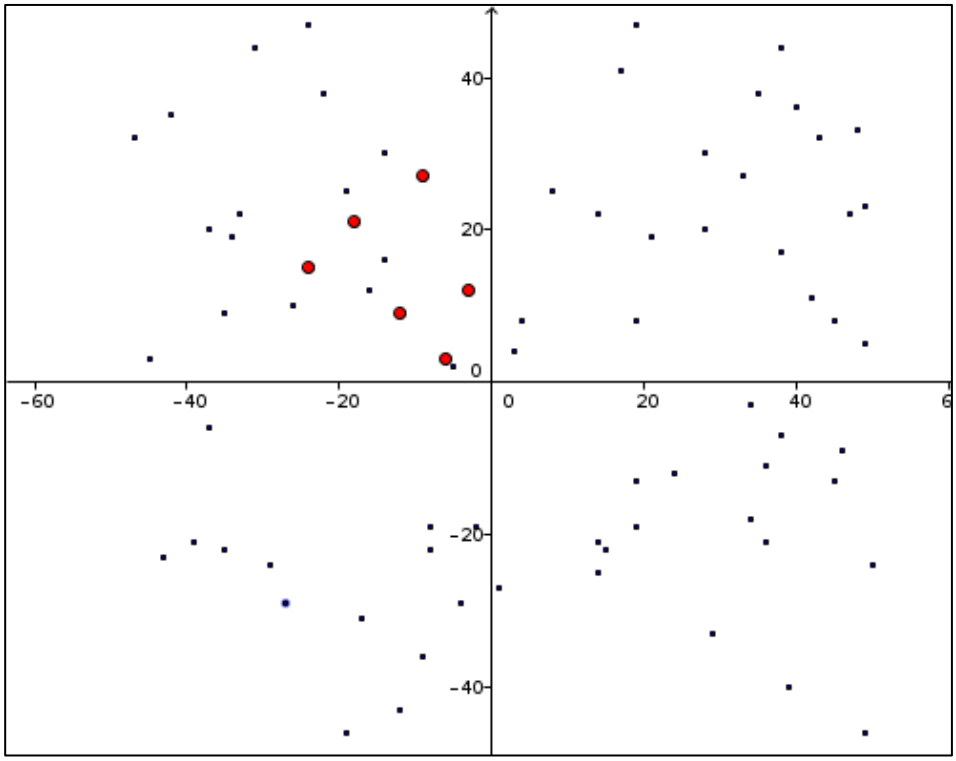


Figura 25 Configuración encontrada

La configuración que se encontró es, con las siguientes coordenadas: (-24,15),(-9,27),(-3,12),(-12,9),(-6,3),(-18,21).

8.3 Pruebas.

Para cada algoritmo que se implementó, se hicieron las pruebas correspondientes para poder determinar su eficacia y el tiempo que se tarda para decir si se encuentra o no la configuración que buscamos en el espacio de búsqueda.

8.3.1 Experimentos.

Las pruebas que se realizaron fueron con configuraciones de distintos tamaños y un espacio de búsqueda de 400 puntos generados de forma aleatoria.

Los 400 puntos de nuestras pruebas fueron generados en cada ejecución de la siguiente forma:

- Las coordenadas x's de nuestros puntos generados aleatoriamente están en el intervalo [-50,50].
- Las coordenadas y's de nuestros puntos generados aleatoriamente están en el intervalo [-50,50].
- La función para calcular los números aleatorios es la función estándar de C, la cual es rand() y no usamos semilla.

- La fórmula utilizada para obtener los números en el intervalo deseado fue la siguiente (1 + rand() modulo 100)-49.
- Los 400 puntos se conforman de la siguiente manera: (400 – número de puntos de la configuración que buscamos).
- Los puntos de la configuración que buscamos se introducen en el espacio de búsqueda manualmente y de forma aleatoria es decir, se van intercalando los puntos de la configuración entre los puntos del espacio.

8.3.2 Resultados de los experimentos.

Se realizaron las pruebas correspondientes y estos fueron los tiempos que se obtuvieron al ejecutar cada uno de los programas.

Tiempos que se tarda el algoritmo en encontrar la configuración sin modificación, en el espacio de búsqueda:

	Experimento 1	Experimento 2	Experimento 3
Configuración	Tiempo(seg)	Tiempo(seg)	Tiempo(seg)
6	0.107486	0.114921	0.179065
10	0.172713	0.180148	0.244292
50	0.181878	0.189313	0.253457

Tiempos que se tarda el algoritmo en encontrar si hay una copia de la configuración con cambios de escala:

	Experimento 1	Experimento 2	Experimento 3
Configuración	Tiempo(seg)	Tiempo(seg)	Tiempo(seg)
6	0.004646	0.012031	0.07617
10	0.198597	0.732463	1.69832
50	0.36969	0.915956	1.925047

Tiempos que se tarda el algoritmo en encontrar si hay una copia de la configuración con cambios de rotación:

	Experimento 1	Experimento 2	Experimento 3
Configuración	Tiempo(seg)	Tiempo(seg)	Tiempo(seg)
6	0.00546	0.012845	0.076989
10	0.029486	0.563379	1.529244
50	0.283478	0.829744	1.838835

Tiempos que se tarda el algoritmo en encontrar si hay una copia de la configuración con cambios de rotación y escala:

	Experimento 1	Experimento 2	Experimento 3
Configuración	Tiempo(seg)	Tiempo(seg)	Tiempo(seg)
6	1.23394	1.240825	1.304987
10	61.765	63.298893	67.26457
50	83.4165	84.96276	90.86441

Posteriormente se realizó las pruebas con un espacio de búsqueda totalmente aleatorio en donde no introducíamos la configuración a buscar, y estos espacios fueron de tamaño 400, 800, 1200 puntos y estos fueron los tiempos registrados:

Tiempos que se tarda el algoritmo en encontrar si hay o no una copia de la configuración:

	Experimento 1	Experimento 2	Experimento 3
Configuración	Tiempo(seg)	Tiempo(seg)	Tiempo(seg)
6	0.165857	0.173292	0.237436
10	0.166912	0.700805	1.666667
50	0.168443	0.719709	1.7258

8.3.3 Análisis de los resultados

Lo que se puede observar claramente en los resultados obtenidos anteriormente es:

- Si el tamaño de la configuración aumenta el tiempo de corrida.
- Si el espacio tiene más puntos el tiempo de corrida

Los tiempos de ejecución pueden cambiar radicalmente dependiendo el lugar en donde se encuentre la pareja de puntos del espacio que corresponden con C_1 y C_2, si se encuentra al principio es menor que si se encuentra en medio, y si se encuentra al final es mayor el tiempo.

Capítulo 9 “Conclusiones”

Con el diseño, implementación y las pruebas de los 4 algoritmos en este proyecto que cumplen satisfactoriamente los objetivos planteados en el capítulo 4, que dan solución al objetivo general y a los objetivos específicos.

Es por eso que nuestras aplicaciones no cuentan con una interfaz gráfica, ya que se busca que a futuro sean módulos de un programa más grande que los utilice, como podría ser el reconocimiento de la EURion Constellation.

Nuestros algoritmos tienen una complejidad computacionalmente del orden $O(kn^3)$,no importando cuál de las cuatro variantes sea considerada

Con las pruebas que se realizaron se puede observar que entre más elementos contenga la configuración y el espacio de puntos, el tiempo que requieren los mismos se va incrementando.

Con toda esta información que se obtuvo de este proyecto puedo decir que el tema de reconocimiento de patrones es muy interesante y atractivo porque se puede utilizar como medida de seguridad para cosas que uno utiliza cotidianamente y me gustaría poder trabajar en otro proyecto que tenga relación con este tema.

Capítulo 10 "Referencias Bibliográficas"

[1]http://www.banxico.org.mx/ayuda/temas-mas-consultados/falsificacion--billetes-moned.html

[2]http://www.banxico.org.mx/divulgacion/billetes-y-monedas/caracteristicas-billetes-mone.html#Caracteristicasdelosbilletes

[3]Markus Kuhn: The EURion constellation. Security Group presentation, Computer Laboratory, University of Cambridge, 8 February 2002.

[4]MitsutakaKatoh, et al.: Image processing device and method for identifying an input image, and copier scanner and printer including same. OmronCorporation, U.S. Patent 5,845,008

[5]P. Torres, "Algoritmo de búsqueda de configuraciones de enlaces ortogonales en dos y tres dimensiones", Proyecto terminal de ingeniería en computación, Universidad Autónoma Metropolitana, D.F, México, 2010.

[6]L. Valerio, "Separación automática de fondo y fenómeno de interés en imágenes tipo", Proyecto terminal de ingeniería en computación, Universidad Autónoma Metropolitana, D.F, México, 2013.

[7]Shih-hsu Chang et al. "Fast algorithm for point pattern matching:invariant to tranlations, rotations and scale changes", Department of Electrical Engineering, National TsingHua University, Hsinchu, Taiwan 300

[8]Lihua Zhang, "Genetic algorithm for affine point patter matching", Department of Automation, Tsinghua University, Beijing 100084, PR China

[9]OswinAichholzer et al. "ComputationalGeometry: Theory and Applications", Institutefor Software Technology, Graz University of Technology, Graz, Austria, Departamento de Matemáticas, Cinvestav, D.F. México, Mexico, Facultad de Ciencias, Universidad Autónoma de San Luis Potosí, San Luis Potosí, Mexico,Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, D.F. México, Mexico, Departament de Matemàtica Aplicada IV, UniversitatPolitècnica de Catalunya, Barcelona, Spain, Instituto de Matemáticas, Universidad Nacional Autónoma de México, D.F. México, Mexico

Capítulo 11 "Apéndice"

Archivo punto.h

```
// esta clase es la base del proyecto ya que en la elaboracion
de el
// se trabajaran con puntos y estos tendran coordenadas del
tipo (x,y)

class punto
{
private:
            // las variables son de tipo entero, porque dan
menos problemas en
            // elcalculo de las operaciones que realizaremos.
      Int coord_x;
      int coord_y;

public:
            //crea el objeto punto con coordenadas (x,y)
      punto(int x = 0,int y = 0);
            //imprime las coordenadas del objeto punto (x,y)
      void imprime();
            //obtiene la coordenada x del objeto punto
      Int get_coord_x();
       //obtiene la coordena y del objeto punto
      Int get_coord_y();
};
```

Archivo punto.cpp

```
#include "punto.h"
#include <iostream>

Using namespace std;

// crea el objeto punto con coordenas (x,y)
punto::punto(int x, int y)
{
      coord_x = x;
      coord_y = y;
}

// imprime las coordenadas (x,y) del objeto punto
Void punto::imprime()
 {
      cout<<coord_x<<","<<coord_y;
}
```

```
// regresa el valor de la coordenada x del objeto punto una vez
obtenido
Int punto::get_coord_x()
{
     return coord_x;
 }

// regresa el valor de la coordenada y del objeto punto una vez
obtenido
Int punto::get_coord_y()
{

     return coord_y;
 }
```

Archivo pareja.h

```
//Esta clase, se creó para poder obtener la distancia y el
ángulo,
//de los objetos puntos, con los cuales trabajaremos en el
proyecto.

#include "punto.h"

class pareja
{
     private:
          punto inicio;
          punto fin;

     public:
     //Crea el objeto pareja con dos objetos puntos, uno
inicial y uno final.
     pareja(punto inicio1, punto fin1 );
     //Calcula la distancia de los objetos punto.
     int distancia();
      //Calcula el angulo que forma la recta que pasa por los
objetos
           //punto, con repecto al eje x.
     double angulo();
};
```

Archivo pareja.cpp

```
#include "pareja.h"
#include "math.h"
```

```
// crea el objeto pareja con dos puntos, uno inicial y uno
final
pareja::pareja(punto inicio1,punto fin1)
{
      inicio = inicio1;
      fin = fin1;
}

//Calcula la distancia en un plano euclidiano de los objetos
punto, uno inicial y uno final
// y devuelve el valor obtenido.

// Nota: se ocupa la distancia al cuadrado, porque asi evitamos
errores de precision,
//       debido a la distancia es la raiz cuadrada y esta tiene
valores decimales,
//       por eso se deja la distancia al cuadrado.
Int pareja::distancia()
{
      double temp;
      int dif_x,dif_y,dist;

      dif_x = (fin.get_coord_x()) - (inicio.get_coord_x());
      dif_y = (fin.get_coord_y()) - (inicio.get_coord_y());

      //ocupamos la distancia al cuadrado para evitar errores de
      precision, ya que la //distancia
      //es la raiz cuadrada y esta nos daria el valor con una
      parte entera y una //decimal, y la parte decimal
            //nos daria problemas de precision.
      temp  =  (  pow(  (double)dif_x,  (double)2)  )  +  (  pow(
(double)dif_y, (double)2) );
      //se transforma de double a int porque la función pow
      ocupa valores double y si //se le asigna a un int
      //nos marca advertencia que habria posible perdida de
      datos, por eso la //tranformamos.
      dist = (int) temp;
      return dist;
}

//Calcula el angulo en un plano euclidiano que forma la recta
que pasa por un punto //inicial
//y final, con respecto al eje "x" y regresa el valor en
radianes.
double pareja::angulo()
{
      double dif_x,dif_y;
      double angu;
```

```
      dif_y = (fin.get_coord_y()) - (inicio.get_coord_y());
      dif_x = (fin.get_coord_x()) - (inicio.get_coord_x());

      //el angulo con respecto al eje x, lo vamos a calcular de
la siguiente manera
      //sacando el arcotangente de la pendiente "m".
      angu = atan2 (dif_y,dif_x);
      return angu;
}
```

Archivo cunas.h

```
//Esta clase es la encargada de decirnos si los puntos que
estamos buscando de la //configuración
//se encuentran unos puntos similares en el espacio de puntos
en donde estamos //buscando,

//Nota: En esta clase agarramos 3 puntos que estan en la
configuracion que //llamamos "bases",
//  y los otros 3 puntos son del espacio en donde estamos
buscando la configuracion y los llamamos
// "explorar" y por eso se ocupan 6 puntos en esta clase.

#include "pareja.h"

class cunas
{
   //los puntos que tienen la palabra base son los puntos de la
   configuracion que //estamos buscando
  //los puntos que tienen la palabra explorar son los puntos
  del espacio en donde //buscaremos la configuracion
  private:

      punto base_fijo1;
      punto base_fijo2;
      punto base_movil;
      punto explorar_fijo1;
      punto explorar_fijo2;
      punto explorar_movil;
      //Nos dice si el giro es a la izquierda,derehca o son
      colineales los objetos //punto de la configuración
      int vuelta_base();
      //Nos dice si el giro es a la izquierda,derehca o son
      colineales los objetos //punto del espcio en donde
      buscamos la configuración
      int vuelta_explora();

     public:
```

```
     //Crea el objeto cunas con los dos puntos fijos de la
     configuracion que //buscamos.
     cunas(punto b_fijo1, punto b_fijo2);
          //Asigna al objeto cunas otro objeto punto de la
configuracion que buscamos.
     Void asigna_base_movil(punto b_movil);
     //Asigna al objeto cunas los objetos puntos del espacio en
     donde queremos //buscar
          //si se encuentra la configuracion.
     Void asigna_explorar_fijos(punto e_fijo1,punto e_fijo2);
     //Asigna al objeto cunas otro objeto punto del espacio en
     donde queremos //buscar
          //si se encuentra la configuracion.
     Void asigna_explorar_movil(punto e_movil);
     //Dice si los 3 puntos base coinciden con los 3 puntos
     explorar, tomando en //cuenta el
         //factor de estiramiento. Da verdadero si coinciden
aún con rotación.
     bool cumple(float factor);
};
```

Archivo cunas.cpp

```
#include"punto.h"
#include"pareja.h"
#include"cunas.h"
#include <iostream>
#include <math.h>

cunas::cunas(punto b_fijo1, punto b_fijo2)
{
     base_fijo1 = b_fijo1;
     base_fijo2 = b_fijo2;
}

//Asigna al objeto cunas otro objeto punto de la configuracion
que buscamos al cual //llamamos
//base_movil
void cunas::asigna_base_movil(punto b_movil)
{
     this -> base_movil = b_movil;
}

//Asigna al objeto cunas los objetos puntos del espacio en
donde queremos buscar
//si se encuentra la configuracion, y a estos objetos puntos le
llamamos explorar_fijos
```

```
void   cunas::asigna_explorar_fijos(punto      e_fijo1,   punto
e_fijo2)
{
     this -> explorar_fijo1 = e_fijo1;
     this -> explorar_fijo2 = e_fijo2;
}

//Asigna al objeto cunas otro objeto punto del espacio en donde
queremos buscar
//si se encuentra la configuracion, y a este punto le llamamos
explorar_movil
void cunas::asigna_explorar_movil(punto e_movil)
{
     this -> explorar_movil = e_movil;
}

//Se encarga de decidir si hay un giro a la izquierda, a la
derehca   o   son   colineales   los   //objetos   punto   de   la
configuracion
int cunas::vuelta_base()
{
    int giro,x1,x2,x3,y1,y2,y3;
    x1       =       base_fijo1.get_coord_x();       y1       =
base_fijo1.get_coord_y();
    x2       =       base_fijo2.get_coord_x();       y2       =
base_fijo2.get_coord_y();
    x3       =       base_movil.get_coord_x();       y3       =
base_movil.get_coord_y();

    giro = ( (x2 - x1)*(y3 - y1) - (y2 - y1)*(x3 - x1) );

    if(giro >0)
    {
      //el giro es a la izquierda
            giro = 1;
    }
    if(giro < 0)
    {
            giro = -1;
    }
     return giro;
}

//Se encarga de decidir si hay un giro a la izquierda, a la
derehca o son colineales los objetos punto del espcio en donde
buscamos la configuracion
int cunas::vuelta_explora()
{
    int giro,x1,x2,x3,y1,y2,y3;
```

```
    x1      =      explorar_fijo1.get_coord_x();      y1      =
explorar_fijo1.get_coord_y();
    x2      =      explorar_fijo2.get_coord_x();      y2      =
explorar_fijo2.get_coord_y();
    x3      =      explorar_movil.get_coord_x();      y3      =
explorar_movil.get_coord_y();

    giro = ( (x2 - x1)*(y3 - y1) - (y2 - y1)*(x3 - x1) );

    if(giro >0)
    {
       //el giro es a la izquierda
            giro = 1;
    }
    if(giro < 0)
    {
            giro = -1;
    }
     return giro;
}

//Dice si los 3 puntos base coinciden con los 3 puntos
explorar, tomando en cuenta el
//factor de estiramiento. Da verdadero si coinciden aún con
rotación.
bool cunas::cumple(float factor)
{
     bool cumple;
     int camino,camino1;
     cumple = false;

     pareja pa1 (base_fijo1,base_fijo2);
     pareja pa2 (explorar_fijo1,explorar_fijo2);
     pareja pa3 (base_fijo2,base_movil);
     pareja pa4 (explorar_fijo2,explorar_movil);

     camino = vuelta_base();
     camino1 = vuelta_explora();

     if(camino1 == camino)
     {
          if( ((pa1.distancia() * factor) == pa2.distancia())
          && ((pa3.distancia() * factor) == pa4.distancia()))
               {
                    cumple = true;
               }
     }
     else
     {
```

```
                cumple = false;
        }

        return cumple;
}
```

Busca normal.cpp

```
#include"punto.h"
#include"pareja.h"
#include"cunas.h"
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <cstdlib>
#include <time.h>
#include <math.h>
#include <sstream>

using namespace std;
int main(int argc, char *argv[])
{

    ///VARIABLES///
    bool movil_found;
    int i,x1,y1,j;
    //"espacio" es el vector en donde se van a guardar los
puntos que se lean
   //del archivo en donde buscaremos la configuracion
     vector <punto> espacio;
    //"configuracion" es el vector en donde se guardaran los
puntos de la configuracion a buscar
    vector <punto> configuracion;
    //  "puntos_configuracion" es el vector en donde se
guardaran los puntos que son la posible configuracion
encontrada
    vector <punto> encontrados;
     string linea;

     char *archivo1,*archivo2,coma;

     archivo1 = argv[1];
     archivo2 = argv[2];
```

```
    if( (argc!=3) || ( (argc == 2) && (argv[1] == "--help")) )
     {
          cout<<endl;cout<<endl;
          cout<<"uso:        busca_normal        <archivo1.txt>
<archivo2.txt>"<<endl;
          cout<<endl;
          cout<<"busca una configuracion de puntos a dentro de
un espacio de puntos"<<endl;
          cout<<endl;cout<<endl;
          cout<<"       El    archivo1    debe    contener    la
configuracion que buscamos"<<endl;
          cout<<"       El archivo2 debe contener el espacio de
busqueda"<<endl;
          cout<<"       Los dos archivos los esperamos con n
numero de lineas, con la forma 'entero,entero'"<<endl;
     return 0;
     }
     //AQUI SE OBTIENEN LOS PUNTOS DE LA CONFIGURACION QUE SE
VA A BUSCAR EN UN ARCHIVO DE TEXTO//
   ifstream MiArchivo (archivo1);
    if (MiArchivo.is_open())
    {
       while(! MiArchivo.eof())
       {
          getline(MiArchivo,linea);
          istringstream iss(linea);

          if(iss >> x1 &&  iss >> coma && iss>> y1 && coma ==
','')
          {

             punto p(x1,y1);

             configuracion.push_back(p); // se va guardando de
uno en
                                         // uno objetos puntos en un
                                         // vector
          } else {
             cout<< "Aviso: Una linea de tu archivo " <<
archivo1 <<  " no cumple con el formato esperado." << endl;
          }
       }
```

```
        MiArchivo.close();
    }
    else
    {
        cout << "No se pudo abrir la configuracion"<<endl;
    }
////////////////////////////////////////////////////////////////
/////////////
    //AQUI SE OBTIENEN LOS PUNTOS EN DONDE SE BUSCARA LA
CONFIGURACION DE UN ARCHIVO DE TEXTO//
         ifstream MiArchivo1 (archivo2);
    if (MiArchivo1.is_open())
    {
       while(! MiArchivo1.eof())
       {
              getline(MiArchivo1,linea);
          istringstream iss(linea);

          if(iss >> x1 &&  iss >> coma && iss>> y1 && coma ==
','')
          {

             punto p(x1,y1);

             espacio.push_back(p); // se va guardando de uno en
                                   // uno objetos puntos en un
                                   // vector
          } else {
             cout<< "Aviso: Una linea de tu archivo " <<
archivo1 <<  " no cumple con el formato esperado." << endl;
          }
       }
       MiArchivo1.close();
    }
    else
    {
        cout << "No se pudo abrir el espacio"<<endl;
    }
    cout << endl;cout << endl;cout << endl;
////////////////////////////////////////////////////////////////
/////////////////////////////////
    if( (configuracion.size() > 2 ) && (espacio.size() > 2) &&
(espacio.size() >= configuracion.size()))
```

```
    {
          cout<<"Estos son los puntos a buscar:"<<endl;
        for (int i=0;i < configuracion.size();i++)
            {
                configuracion[i].imprime();
                cout<<"   ";
            }

           cout<<endl;cout<<endl;
        cout << "El espacio en donde buscamos tiene este numero
de puntos:"<< espacio.size() <<endl;
        cout<<endl;cout<<endl;
        cout <<"BUSCA SI ESTA LA CONFIGURACION ORIGINAL"<<endl;

        cunas cu (configuracion[0],configuracion[1]);
        for(int  j=0;j  <  espacio.size();j++)  // Se escoge el
primer  punto  del  espacio  en  donde  vamos  a  buscar  la
configuracion
        {
            encontrados.push_back(espacio[j]);//guardamos    el
primer punto que puede ser de la configuracion que buscamos
            for(int k = 0; k < espacio.size() ;k++)// Se escoge
el  segundo  punto  del  espacio  en  donde  vamos  a  buscar  la
configuracion
            {
                if(k != j) // no se hace nada si los puntos son
los mismos
                {
                    pareja                                  pa1
(configuracion[0],configuracion[1]);  //se  crean  las  parejas
correspondientes  de  los  puntos  de  la  configuracion  y  del
espacio
              pareja pa2 (espacio[j],espacio[k]);// para poder
realizar nuestras operaciones correspondientes

                        if(        ((pa1.distancia())        ==
(pa2.distancia()))  &&  ((pa1.angulo())  ==  (pa2.angulo()))) //se
comparan si las distancias y angulos son las mismas
                        {

cu.asigna_explorar_fijos(espacio[j],espacio[k]);//una  vez  que
corroboramos  que  los  puntos  pueden  ser  candidatos  de  la
```

```
configuracion que buscamos se guardan en un objeto cunas para
poder utilizarlos mas adelante.

encontrados.push_back(espacio[k]);//guardamos el segundo punto
que puede ser de la configuracion que buscamos

                        for(int h=2; h <
configuracion.size();h++)
                        {
                            movil_found = false;//utilizamos
una bandera para decir si el punto que estamos buscando cumple
o no con las caracteristicas que necesitamos

cu.asigna_base_movil(configuracion[h]);//asignamos el siguiente
punto de la configuracion y lo guardamos en el objeto cunas
para utilizarlo despues

                            for(int l=0;l < espacio.size();l++)
                            {
                                if((l!=j) && (l!=k))
                                {

cu.asigna_explorar_movil(espacio[l]);//asignamos un punto que
no sea el punto j o el punto k del espacio en la cuna para
poder utilizarlo despues

                                    if(cu.cumple(1.0))//cumple
se encarga de hacer las operaciones necesarias y corroborar si
el punto l del espacio cumple con las caracteristicas que se
necesitan
                                    {
                                        movil_found = true;//si
cumple el punto l con las caracteristicas aporpiadas

encontrados.push_back(espacio[l]);//guardamos el punto l en el
vector que contiene los puntos que son candidatos a ser la
configuracion que buscamos
                                        break;
                                    }
                                }
                            }

                            if(movil_found == false)
```

```
                                    break; //el punto l no cumple
con las caracteristicas y se sale del ciclo
                                }

                            if(encontrados.size()                    ==
configuracion.size())
                                    break;//si     el     espacio     y
configuracion tienen el mismo tamaño se sale
                            encontrados.clear();//se limpia el vector
en donde estamos guardando los puntos para escoger un nuevo
punto k que pueda ser de la configuracion que buscamos
                            encontrados.push_back(espacio[j]);//se
guarda el punto j en el vector porque solo estamos cambiando de
punto k
                        }
                    }
                }

                if (encontrados.size() == configuracion.size())
                    break;//si el espacio y configuracion son del
mismo tamaño se sale del ciclo
                encontrados.clear();//Se  limpia  el  vector  para
empezar  a  buscar  unos  nuevos  puntos  que  cumplan  con  la
configuracion
            }
            if( encontrados.size() == configuracion.size())
             {
                cout<<"Si se encuentra la configuracion y estos son
su puntos:"<<endl;
                for (int i=0;i < encontrados.size();i++)
                {
                    encontrados[i].imprime();
                    cout<<"   ";
                }
              cout <<endl; cout <<endl;
            }
            else
            {
                cout<<"No se encuentra la configuracion"<<endl;
                cout<<""<<endl;cout<<""<<endl;
            }
```

```
      }
     else
     {
           if(configuracion.size() < 3 )
           {
                cout<<"La configuracion no es de tamaño 3 o
mas"<<endl;
           }
           if(espacio.size() < 3)
           {
                cout<<"El espacio no es de tamaño 3 o
mas"<<endl;
           }
           if(espacio.size() < configuracion.size())
           {
             cout<<"La configuracion es de mayor tamaño que el
espacio"<<endl;
         }
         cout<<endl;
     }

     return EXIT_SUCCESS;
}
```

Busca aumento.cpp

```
#include"punto.h"
#include"pareja.h"
#include"cunas.h"
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <cstdlib>
#include <time.h>
#include <math.h>
#include <sstream>

using namespace std;
int main(int argc, char *argv[])
{
```

```
    ///VARIABLES///
    bool movil_found;
    int i,x1,y1,j;
    //"espacio" es el vector en donde se van a guardar los
puntos que se lean
   //del archivo en donde buscaremos la configuracion
     vector <punto> espacio;
    //"configuracion" es el vector en donde se guardaran los
puntos de la configuracion a buscar
    vector <punto> configuracion;
    //  "puntos_configuracion" es el vector en donde se
guardaran los puntos que son la posible configuracion
encontrada
    vector <punto> encontrados;
     string linea;

     char *archivo1,*archivo2,coma;

     archivo1 = argv[1];
     archivo2 = argv[2];

    if( (argc!=3) || ( (argc == 2) && (argv[1] == "--help")) )
     {
          cout<<endl;cout<<endl;
          cout<<"uso:        busca_normal        <archivo1.txt>
<archivo2.txt>"<<endl;
          cout<<endl;
          cout<<"busca una configuracion de puntos a dentro de
un espacio de puntos"<<endl;
          cout<<endl;cout<<endl;
          cout<<"       El    archivo1    debe    contener    la
configuracion que buscamos"<<endl;
          cout<<"       El archivo2 debe contener el espacio de
busqueda"<<endl;
          cout<<"       Los dos archivos los esperamos con n
numero de lineas, con la forma 'entero,entero'"<<endl;
     return 0;
     }
     //AQUI SE OBTIENEN LOS PUNTOS DE LA CONFIGURACION QUE SE
VA A BUSCAR EN UN ARCHIVO DE TEXTO//
   ifstream MiArchivo (archivo1);
    if (MiArchivo.is_open())
    {
```

```
        while(! MiArchivo.eof())
        {
           getline(MiArchivo,linea);
           istringstream iss(linea);

           if(iss >> x1 &&  iss >> coma && iss>> y1 && coma ==
','')
           {

              punto p(x1,y1);

              configuracion.push_back(p); // se va guardando de
uno en
                                          // uno objetos puntos en un
                                          // vector
           } else {
              cout<< "Aviso: Una linea de tu archivo " <<
archivo1 <<  " no cumple con el formato esperado." << endl;
           }
        }
        MiArchivo.close();
     }
     else
     {
         cout << "No se pudo abrir la configuracion"<<endl;
     }
////////////////////////////////////////////////////////////////
/////////////
     //AQUI SE OBTIENEN LOS PUNTOS EN DONDE SE BUSCARA LA
CONFIGURACION DE UN ARCHIVO DE TEXTO//
          ifstream MiArchivo1 (archivo2);
     if (MiArchivo1.is_open())
     {
        while(! MiArchivo1.eof())
        {
              getline(MiArchivo1,linea);
           istringstream iss(linea);

           if(iss >> x1 &&  iss >> coma && iss>> y1 && coma ==
','')
           {

              punto p(x1,y1);
```

```
                espacio.push_back(p); // se va guardando de uno en
                                      // uno objetos puntos en un
                                      // vector
            } else {
                cout<< "Aviso: Una linea de tu archivo " <<
archivo1 <<  " no cumple con el formato esperado." << endl;
            }
        }
        MiArchivo1.close();
    }
    else
    {
        cout << "No se pudo abrir el espacio"<<endl;
    }
    cout << endl;cout << endl;cout << endl;
////////////////////////////////////////////////////////////////
////////////////////////////////
    if( (configuracion.size() > 2 ) && (espacio.size() > 2) &&
(espacio.size() >= configuracion.size()))
    {
          cout<<"Estos son los puntos a buscar:"<<endl;
        for (int i=0;i < configuracion.size();i++)
            {
                configuracion[i].imprime();
                cout<<"   ";
            }

           cout<<endl;cout<<endl;
        cout << "El espacio en donde buscamos tiene este numero
de puntos:"<< espacio.size() <<endl;
        cout<<endl;cout<<endl;
        cout    <<"BUSCA    SI    ESTA    LA    CONFIGURACION
AUMENTADA"<<endl;

        cunas cu1 (configuracion[0],configuracion[1]);
        for(int  j=0;j  <  espacio.size();j++)  // Se escoge el
primer  punto  del  espacio  en  donde  vamos  a  buscar  la
configuracion
        {
            encontrados.push_back(espacio[j]);//guardamos    el
primer punto que puede ser de la configuracion que buscamos
```

```
            for(int k = 0; k < espacio.size() ;k++)// Se escoge
el segundo punto del espacio en donde vamos a buscar la
configuracion
            {
                if(k != j) // no se hace nada si los puntos son
los mismos
                {
                    pareja                                  pa11
(configuracion[0],configuracion[1]);//se crean las parejas
correspondientes de los puntos de la configuracion y del
espacio
                        pareja pa21 (espacio[j],espacio[k]);//
para poder realizar nuestras operaciones correspondientes

                        if( (pa11.angulo()) == (pa21.angulo())
)//se compara si los angulos son los mismos
                        {
                        factor   =   (   (pa21.distancia())   /
(pa11.distancia()) );//obtienen el factor de aumento

cu1.asigna_explorar_fijos(espacio[j],espacio[k]);//una vez que
corroboramos que los puntos pueden ser candidatos de la
configuracion que buscamos se guardan en un objeto cunas para
poder utilizarlos mas adelante.

encontrados.push_back(espacio[k]);//guardamos el segundo punto
que puede ser de la configuracion que buscamos

                        for(int         h=2;         h         <
configuracion.size();h++)
                        {
                            movil_found   =   false;//utilizamos
una bandera para decir si el punto que estamos buscando cumple
o no con las caracteristicas que necesitamos

cu1.asigna_base_movil(configuracion[h]);//asignamos          el
siguiente punto de la configuracion y lo guardamos en el objeto
cunas para utilizarlo despues

                            for(int l=0;l < espacio.size();l++)
                            {
                                if((l!=j) && (l!=k))
                                {
```

```
cu1.asigna_explorar_movil(espacio[l]);//asignamos un punto que
no sea el punto j o el punto k del espacio en la cuna para
poder utilizarlo despues


if(cu1.cumple(factor))//cumple se encarga de hacer las
operaciones necesarias y corroborar si el punto l del espacio
cumple con las caracteristicas que se necesitan con un factor
de aumento
                                        {
                                            movil_found = true;//si
cumple el punto l con las caracteristicas aporpiadas

encontrados.push_back(espacio[l]);//guardamos el punto l en el
vector que contiene los puntos que son candidatos a ser la
configuracion que buscamos
                                            break;
                                        }
                                    }
                                }

                                if(movil_found == false)
                                    break;//el punto no cumple con
las caracteristicas y se sale del ciclo
                            }

                        if(encontrados.size()                    ==
configuracion.size())
                             break;//si     el     espacio     y
configuracion tienen el mismo tamaño se sale
                        encontrados.clear();//se limpia el vector
en donde estamos guardando los puntos para escoger un nuevo
punto k que pueda ser de la configuracion que buscamos
                        encontrados.push_back(espacio[j]);//se
guarda el punto j en el vector porque solo estamos cambiando de
punto k
                    }
                }
            }

            if (encontrados.size() == configuracion.size())
```

```
                break;//si el espacio y configuracion tienen el
mismo tamaño se sale
            encontrados.clear();//Se  limpia  el  vector  para
empezar  a  buscar  unos  nuevos  puntos  que  cumplan  con  la
configuracion
        }

        if( encontrados.size() == configuracion.size())
         {
             cout<<"Si se encuentra la configuracion y estos son
su puntos:"<<endl;
             for (int i=0;i < encontrados.size();i++)
             {
                 encontrados[i].imprime();
                 cout<<"   ";
             }
           cout <<endl; cout <<endl;
         }
         else
         {
             cout<<"No se encuentra la configuracion"<<endl;
             cout<<""<<endl;cout<<""<<endl;
         }



      }
     else
     {
           if(configuracion.size() < 3 )
           {
                cout<<"La  configuracion  no  es  de  tamaño  3  o
mas"<<endl;
           }
           if(espacio.size() < 3)
           {
                cout<<"El  espacio  no  es  de  tamaño  3  o
mas"<<endl;
           }
           if(espacio.size() < configuracion.size())
           {
             cout<<"La configuracion es de mayor tamaño que el
espacio"<<endl;
```

```
        }
        cout<<endl;
    }

    return EXIT_SUCCESS;
}
```

Busca_rotada.cpp

```
#include"punto.h"
#include"pareja.h"
#include"cunas.h"
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <cstdlib>
#include <time.h>
#include <math.h>
#include <sstream>

using namespace std;
int main(int argc, char *argv[])
{

    ///VARIABLES///
    bool movil_found;
    int i,x1,y1,j;
    //"espacio" es el vector en donde se van a guardar los
puntos que se lean
   //del archivo en donde buscaremos la configuracion
     vector <punto> espacio;
    //"configuracion" es el vector en donde se guardaran los
puntos de la configuracion a buscar
    vector <punto> configuracion;
    //  "puntos_configuracion" es el vector en donde se
guardaran los puntos que son la posible configuracion
encontrada
    vector <punto> encontrados;
     string linea;

     char *archivo1,*archivo2,coma;
```

```
     archivo1 = argv[1];
     archivo2 = argv[2];

    if( (argc!=3) || ( (argc == 2) && (argv[1] == "--help")) )
     {
          cout<<endl;cout<<endl;
          cout<<"uso:        busca_normal        <archivo1.txt>
<archivo2.txt>"<<endl;
          cout<<endl;
          cout<<"busca una configuracion de puntos a dentro de
un espacio de puntos"<<endl;
          cout<<endl;cout<<endl;
          cout<<"       El    archivo1    debe    contener    la
configuracion que buscamos"<<endl;
          cout<<"       El archivo2 debe contener el espacio de
busqueda"<<endl;
          cout<<"       Los dos archivos los esperamos con n
numero de lineas, con la forma 'entero,entero'"<<endl;
     return 0;
     }
     //AQUI SE OBTIENEN LOS PUNTOS DE LA CONFIGURACION QUE SE
VA A BUSCAR EN UN ARCHIVO DE TEXTO//
   ifstream MiArchivo (archivo1);
    if (MiArchivo.is_open())
    {
       while(! MiArchivo.eof())
       {
          getline(MiArchivo,linea);
          istringstream iss(linea);

          if(iss >> x1 &&  iss >> coma && iss>> y1 && coma ==
','')
          {

             punto p(x1,y1);

             configuracion.push_back(p); // se va guardando de
uno en
                                         // uno objetos puntos en un
                                         // vector
          } else {
             cout<< "Aviso: Una linea de tu archivo " <<
archivo1 <<  " no cumple con el formato esperado." << endl;
```

```
            }
        }
        MiArchivo.close();
    }
    else
    {
        cout << "No se pudo abrir la configuracion"<<endl;
    }
////////////////////////////////////////////////////////////////
/////////////
    //AQUI SE OBTIENEN LOS PUNTOS EN DONDE SE BUSCARA LA
CONFIGURACION DE UN ARCHIVO DE TEXTO//
         ifstream MiArchivo1 (archivo2);
    if (MiArchivo1.is_open())
    {
        while(! MiArchivo1.eof())
        {
               getline(MiArchivo1,linea);
           istringstream iss(linea);

           if(iss >> x1 &&  iss >> coma && iss>> y1 && coma ==
','')
           {

              punto p(x1,y1);

              espacio.push_back(p); // se va guardando de uno en
                                    // uno objetos puntos en un
                                    // vector
           } else {
              cout<< "Aviso: Una linea de tu archivo " <<
archivo1 <<  " no cumple con el formato esperado." << endl;
           }
        }
        MiArchivo1.close();
    }
    else
    {
        cout << "No se pudo abrir el espacio"<<endl;
    }
    cout << endl;cout << endl;cout << endl;
////////////////////////////////////////////////////////////////
/////////////////////////////////
```

```
    if( (configuracion.size() > 2 ) && (espacio.size() > 2) &&
(espacio.size() >= configuracion.size()))
    {
          cout<<"Estos son los puntos a buscar:"<<endl;
        for (int i=0;i < configuracion.size();i++)
            {
                configuracion[i].imprime();
                cout<<"    ";
            }

           cout<<endl;cout<<endl;
        cout << "El espacio en donde buscamos tiene este numero
de puntos:"<< espacio.size() <<endl;
        cout<<endl;cout<<endl;
        cout <<"BUSCA SI ESTA LA CONFIGURACION ROTADA"<<endl;

        cunas cu2 (configuracion[0],configuracion[1]);
        for(int  j=0;j  <  espacio.size();j++)  // Se escoge el
primer  punto  del  espacio  en  donde  vamos  a  buscar  la
configuracion
        {
            encontrados.push_back(espacio[j]);//guardamos    el
primer punto que puede ser de la configuracion que buscamos
            for(int k = 0; k < espacio.size() ;k++)// Se escoge
el  segundo  punto  del  espacio  en  donde  vamos  a  buscar  la
configuracion
            {
                if(k != j) // no se hace nada si los puntos son
los mismos
                {
                    pareja                                  pa12
(configuracion[0],configuracion[1]);//se  crean  las  parejas
correspondientes  de  los  puntos  de  la  configuracion  y  del
espacio
                        pareja  pa22  (espacio[j],espacio[k]);//
para poder realizar nuestras operaciones correspondientes con
los puntos

                        angulo    =    (    (pa22.angulo()    -
pa12.angulo()) * 180 ) / M_PI;                  //calcula  el
angulo de rotamiento con respecto al eje x
```

```
                        if(         (pa12.distancia())         ==
(pa22.distancia()) ) //se compara si las distancias son iguales
                        {

cu2.asigna_explorar_fijos(espacio[j],espacio[k]);//una vez que
corroboramos que los puntos pueden ser candidatos de la
configuracion que buscamos se guardan en un objeto cunas para
poder utilizarlos mas adelante.

encontrados.push_back(espacio[k]);//guardamos el segundo punto
que puede ser de la configuracion que buscamos

                        for(int          h=2;          h          <
configuracion.size();h++)
                        {
                            movil_found   =   false;//utilizamos
una bandera para decir si el punto que estamos buscando cumple
o no con las caracteristicas que necesitamos

cu2.asigna_base_movil(configuracion[h]);//asignamos          el
siguiente punto de la configuracion y lo guardamos en el objeto
cunas para utilizarlo despues

                            for(int l=0;l < espacio.size();l++)
                            {
                                if((l!=j) && (l!=k))
                                {

cu2.asigna_explorar_movil(espacio[l]);//asignamos un punto que
no sea el punto j o el punto k del espacio en la cuna para
poder utilizarlo despues

                                    if(cu2.cumple(1.0))//cumple
se encarga de hacer las operaciones necesarias y corroborar si
el punto l del espacio cumple con las caracteristicas que se
necesitan
                                    {
                                        movil_found = true;//si
cumple el punto con las caracteristicas aporpiadas

encontrados.push_back(espacio[l]);//guardamos el punto l en el
vector que contiene los puntos que son candidatos a ser la
configuracion que buscamos
```

```
                                        break;
                                    }
                                }
                            }

                            if(movil_found == false)
                                break;//el punto l no cumple
con las caracteristicas y se sale del ciclo
                            }

                        if(encontrados.size()                    ==
configuracion.size())
                             break;//si     el     espacio     y
configuracion tienen el mismo tamaño se sale
                        encontrados.clear();//se limpia el vector
en donde estamos guardando los puntos para escoger un nuevo
punto k que pueda ser de la configuracion que buscamos
                        encontrados.push_back(espacio[j]);//se
guarda el punto j en el vector porque solo estamos cambiando de
punto k
                    }
                }
            }


            if (encontrados.size() == configuracion.size())
                break;//si el espacio y configuracion tienen el
mismo tamaño se sale
            encontrados.clear();//Se  limpia  el  vector  para
empezar  a  buscar  unos  nuevos  puntos  que  cumplan  con  la
configuracion
        }

        if( encontrados.size() == configuracion.size())
         {
             cout<<"Si se encuentra la configuracion y estos son
su puntos:"<<endl;
             for (int i=0;i < encontrados.size();i++)
             {
                 encontrados[i].imprime();
                 cout<<"   ";
             }
          cout <<endl; cout <<endl;
```

```
        }
        else
        {
            cout<<"No se encuentra la configuracion"<<endl;
            cout<<""<<endl;cout<<""<<endl;
        }



     }
    else
    {
          if(configuracion.size() < 3 )
          {
               cout<<"La configuracion no es de tamaño 3 o
mas"<<endl;
          }
          if(espacio.size() < 3)
          {
               cout<<"El espacio no es de tamaño 3 o
mas"<<endl;
          }
          if(espacio.size() < configuracion.size())
          {
            cout<<"La configuracion es de mayor tamaño que el
espacio"<<endl;
        }
        cout<<endl;
    }

    return EXIT_SUCCESS;
}
```

Busca_rotada_aumento.cpp

```
#include"punto.h"
#include"pareja.h"
#include"cunas.h"
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <cstdlib>
```

```
#include <time.h>
#include <math.h>
#include <sstream>

using namespace std;
int main(int argc, char *argv[])
{

    ///VARIABLES///
    bool movil_found;
    int i,x1,y1,j;
    //"espacio" es el vector en donde se van a guardar los
puntos que se lean
   //del archivo en donde buscaremos la configuracion
     vector <punto> espacio;
    //"configuracion" es el vector en donde se guardaran los
puntos de la configuracion a buscar
    vector <punto> configuracion;
    //  "puntos_configuracion" es el vector en donde se
guardaran los puntos que son la posible configuracion
encontrada
    vector <punto> encontrados;
     string linea;

     char *archivo1,*archivo2,coma;

     archivo1 = argv[1];
     archivo2 = argv[2];

    if( (argc!=3) || ( (argc == 2) && (argv[1] == "--help")) )
     {
          cout<<endl;cout<<endl;
          cout<<"uso:        busca_normal        <archivo1.txt>
<archivo2.txt>"<<endl;
          cout<<endl;
          cout<<"busca una configuracion de puntos a dentro de
un espacio de puntos"<<endl;
          cout<<endl;cout<<endl;
          cout<<"       El    archivo1    debe    contener    la
configuracion que buscamos"<<endl;
          cout<<"       El archivo2 debe contener el espacio de
busqueda"<<endl;
```

```
          cout<<"       Los dos archivos los esperamos con n
numero de lineas, con la forma 'entero,entero'"<<endl;
     return 0;
     }
     //AQUI SE OBTIENEN LOS PUNTOS DE LA CONFIGURACION QUE SE
VA A BUSCAR EN UN ARCHIVO DE TEXTO//
   ifstream MiArchivo (archivo1);
    if (MiArchivo.is_open())
    {
       while(! MiArchivo.eof())
       {
          getline(MiArchivo,linea);
          istringstream iss(linea);

          if(iss >> x1 &&  iss >> coma && iss>> y1 && coma ==
','')
          {

             punto p(x1,y1);

             configuracion.push_back(p); // se va guardando de
uno en
                                         // uno objetos puntos en un
                                         // vector
          } else {
             cout<< "Aviso: Una linea de tu archivo " <<
archivo1 <<  " no cumple con el formato esperado." << endl;
          }
       }
       MiArchivo.close();
    }
    else
    {
        cout << "No se pudo abrir la configuracion"<<endl;
    }
////////////////////////////////////////////////////////////////
/////////////
    //AQUI SE OBTIENEN LOS PUNTOS EN DONDE SE BUSCARA LA
CONFIGURACION DE UN ARCHIVO DE TEXTO//
         ifstream MiArchivo1 (archivo2);
    if (MiArchivo1.is_open())
    {
       while(! MiArchivo1.eof())
```

```
        {
              getline(MiArchivo1,linea);
           istringstream iss(linea);

           if(iss >> x1 &&  iss >> coma && iss>> y1 && coma ==
','')
           {

              punto p(x1,y1);

              espacio.push_back(p); // se va guardando de uno en
                                    // uno objetos puntos en un
                                    // vector
           } else {
              cout<< "Aviso: Una linea de tu archivo " <<
archivo1 <<  " no cumple con el formato esperado." << endl;
           }
        }
        MiArchivo1.close();
     }
     else
     {
         cout << "No se pudo abrir el espacio"<<endl;
     }
     cout << endl;cout << endl;cout << endl;
////////////////////////////////////////////////////////////////
////////////////////////////////
     if( (configuracion.size() > 2 ) && (espacio.size() > 2) &&
(espacio.size() >= configuracion.size()))
     {
           cout<<"Estos son los puntos a buscar:"<<endl;
         for (int i=0;i < configuracion.size();i++)
             {
                 configuracion[i].imprime();
                 cout<<"   ";
             }

            cout<<endl;cout<<endl;
         cout << "El espacio en donde buscamos tiene este numero
de puntos:"<< espacio.size() <<endl;
         cout<<endl;cout<<endl;
         cout  <<"BUSCA SI ESTA LA CONFIGURACION AUMENTADA Y
ROTADA"<<endl;
```

```
        cunas cu3 (configuracion[0],configuracion[1]);
        for(int  j=0;j  <  espacio.size();j++)  // Se escoge el
primer  punto  del  espacio  en  donde  vamos  a  buscar  la
configuracion
        {
            encontrados.push_back(espacio[j]);//guardamos    el
primer punto que puede ser de la configuracion que buscamos
            for(int k = 0; k < espacio.size() ;k++)// Se escoge
el  segundo  punto  del  espacio  en  donde  vamos  a  buscar  la
configuracion
            {
                if(k != j) // no se hace nada si los puntos son
los mismos
                {
                        pareja
pa13(configuracion[0],configuracion[1]);//se crean las parejas
correspondientes  de  los  puntos  de  la  configuracion  y  del
espacio
                                pareja                          pa23
(espacio[j],espacio[k]);//   para   poder   realizar   nuestras
operaciones correspondientes
                    angulo  =  (  (pa23.angulo()  -  pa13.angulo())  *
180 ) / M_PI;//calcula el angulo de rotamiento
                                factor  =  (  (pa23.distancia())  /
(pa13.distancia()) ); //se obtiene el factor de aumento

cu3.asigna_explorar_fijos(espacio[j],espacio[k]);//una  vez  que
corroboramos  que  los  puntos  pueden  ser  candidatos  de  la
configuracion que buscamos se guardan en un objeto cunas para
poder utilizarlos mas adelante.

encontrados.push_back(espacio[k]);//guardamos  el  segundo  punto
que puede ser de la configuracion que buscamos

                        for(int         h=2;         h         <
configuracion.size();h++)
                        {
                            movil_found   =   false;//utilizamos
una bandera para decir si el punto que estamos buscando cumple
o no con las caracteristicas que necesitamos

cu3.asigna_base_movil(configuracion[h]);//asignamos          el
```

```
siguiente punto de la configuracion y lo guardamos en el objeto
cunas para utilizarlo despues

                                for(int l=0;l < espacio.size();l++)
                                {
                                    if((l!=j) && (l!=k))
                                    {

cu3.asigna_explorar_movil(espacio[l]);//asignamos un punto que
no sea el punto j o el punto k del espacio en la cuna para
poder utilizarlo despues


if(cu3.cumple(factor))//cumple se encarga de hacer las
operaciones necesarias y corroborar si el punto l del espacio
cumple con las caracteristicas que se necesitan con un factor
de aumento
                                        {
                                            movil_found = true;//si
cumple el punto l con las caracteristicas aporpiadas

encontrados.push_back(espacio[l]);//guardamos el punto l en el
vector que contiene los puntos que son candidatos a ser la
configuracion que buscamos
                                            break;
                                        }
                                    }
                                }

                                if(movil_found == false)
                                    break;//el punto l no cumple
con las caracteristicas y se sale del ciclo
                                }

                            if(encontrados.size()                    ==
configuracion.size())
                                break;//si el espacio y
configuracion tienen el mismo tamaño se sale
                            encontrados.clear();//se limpia el vector
en donde estamos guardando los puntos para escoger un nuevo
punto k que pueda ser de la configuracion que buscamos
```

```
                        encontrados.push_back(espacio[j]);//se
guarda el punto j en el vector porque solo estamos cambiando de
punto k

                }
            }


            if (encontrados.size() == configuracion.size())
                break;//si el espacio y configuracion tienen el
mismo tamaño se sale
            encontrados.clear();//Se  limpia  el  vector  para
empezar  a  buscar  unos  nuevos  puntos  que  cumplan  con  la
configuracion
        }

        if( encontrados.size() == configuracion.size())

         {
             cout<<"Si se encuentra la configuracion y estos son
su puntos:"<<endl;
             for (int i=0;i < encontrados.size();i++)
             {
                 encontrados[i].imprime();
                 cout<<"   ";
             }
           cout <<endl; cout <<endl;
         }
         else
         {
             cout<<"No se encuentra la configuracion"<<endl;
             cout<<""<<endl;cout<<""<<endl;
         }



      }
    else
    {
          if(configuracion.size() < 3 )
          {
              cout<<"La  configuracion  no  es  de  tamaño  3  o
mas"<<endl;
```

```
            }
            if(espacio.size() < 3)
            {
                 cout<<"El espacio no es de tamaño 3 o
mas"<<endl;
            }
            if(espacio.size() < configuracion.size())
            {
              cout<<"La configuracion es de mayor tamaño que el
espacio"<<endl;
         }
         cout<<endl;
     }

     return EXIT_SUCCESS;
}
```

Capítulo 12 “Entregables”

El CD que se entregara contiene lo siguiente:

- Reporte conteniendo toda la investigación teórica desarrollada en el proyecto.
- Código fuente de todos los programas desarrollados.