MEMORIA TRABAJO FIN DE GRADO

IMPLEMENTACIÓN DE ALGORITMO DE SCAN-MATCHING BASADO EN CLONALG

Autor: Laura Carrasco Payo

Tutor: Fernando Martín Monar

UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

GRADO INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

AGRADECIMIENTOS

A mi familia, y en especial a mis padres, Jose y Toñi, por su apoyo incondicional, especialmente en los malos momentos, sus sabios consejos y su inagotable paciencia a lo largo de todos estos años.

A mi tutor, Fernando, por el conocimiento y la ayuda que me ha aportado para realizar este trabajo.

A todos los profesores y docentes que de una forma u otra han contribuido a que adquiera una serie de conocimientos y aptitudes en estos años.

Y a mis compañeros y amigos que me han apoyado y siempre han estado ahí.

RESUMEN

La estimación de la posición de robots ha sido una de las ramas de la robótica más estudiada en los últimos años. Es importante para la autonomía de un robot conocer el entorno y posicionarse en él. Para ello se han dado múltiples soluciones al conocido problema del Simultaneous Localization and Mapping (SLAM), que está íntimamente ligado al scan matching o alineación de pares de imágenes.

Para realizar un matching entre imágenes es necesario sacar características del entorno y hacer una asociación de datos correctamente, lo cual es una compleja tarea. Gracias al desarrollo de los sensores en la actualidad se ha conseguido obtener más características del entorno que pueden resultar de gran utilidad para resolver estos problemas, como las características del color.

Por otro lado el progreso y la evolución en el mundo de la inteligencia artificial, IA, ha hecho que se desarrollen potentes algoritmos aplicables a múltiples ámbitos y que aportan soluciones muy satisfactorias y eficientes. Hay una gran variedad de algoritmos, entre ellos están el Differential Evolution (DE) y el Clonal Selection Algorithm (CLONALG). Ambos son utilizados para dar soluciones a múltiples problemas computacionales, y la optimización es uno de ellos.

Este problema de optimización se puede aplicar para realizar el matching o alineamiento entre imágenes. Para ello se debe utilizar una función objetivo o función de coste que se optimizará utilizando un algoritmo. En este trabajo para realizar esta función se va a emplear el CLONALG, un algoritmo evolutivo basado en el principio de selección clonal. Este algoritmo es utilizado en múltiples tareas entre las que destacan el reconocimiento de patrones o la optimización.

ABSTRACT

Estimating the position of robots has been widely studied during the last decades. To improve the autonomy of a mobilerobot, it is important to obtain an accurate estimate of the robot's location. The Simultaneous Localization and Mapping problem, which is closely related to this task, has multiple solutions and it is closely linked to the scan matching technique.

To perform an adequate matching between images or depth scans, it is necessary to have the proper information and it is important to remove unnecessary data, which is a complex task. In the last years, the development of new sensors has provided more detailed information (colour and depth) about images.

On the other hand, the constant developments and the evolution in the artificial intelligence research field has resulted inpowerful algorithms that can be applied to multiple tasks. There are multiple methods that can provide satisfactory and efficient solutions to many optimization problems. The evolutionary methods are among them. Two examples are the Differential Evolution and the CLONALG algorithms.

The scan matching task can be viewed as an optimization problem. A fitness function can be defined to estimate the quality of the matching process. If the fitness function is implemented in an adequate way, an optimization method can be applied to obtain the solution to the problem. The goal of this work is to implement a CLONALG-based scan matching algorithm. The efficiency will be demonstrated in the experimental results.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE FÓRMULAS

1. INTRODUCCIÓN

Esta memoria describe el desarrollo de un TFG (Trabajo Fin de Grado) en el que se implementa el algoritmo evolutivo, CLONALG, para realizar el matching, o alineación entre imágenes. Este TFG está basado en el proyecto similar desarrollado por Fernando Marín Monar [1] en el que se utiliza otro algoritmo evolutivo para alcanzar la misma solución al mismo problema. Este proyecto utiliza como algoritmo el DE.

El algoritmo desarrollado en este trabajo es el CLONALG, que forma parte de los sistemas inmunológicos artificiales. Su funcionamiento se basa en principios que aparecen en el sistema inmunológico humano. Su aplicación se extiende a varios ámbitos y se utiliza para resolver múltiples problemas pero los más destacados son el reconocimiento de patrones y la optimización. En este trabajo se va a utilizar para resolver este último problema, el de optimización. [2]

Mediante la optimización de funciones se trata de buscar un valor que las minimice o maximice, dependiendo del problema y la solución que se quiera alcanzar. Para ello se parte de una población, o conjunto de individuos, la cual se somete a múltiples cambios y modificaciones de tal forma que al final se consiga seleccionar los buenos candidatos y descartar los que dan peores soluciones.

En definitiva, en este trabajo se va a resolver el problema del scan matching, o solapamiento de imágenes, utilizando un algoritmo evolutivo, el CLONALG, que optimizará una función objetivo o función de coste que representa la alineación entre nubes de puntos de un par de imágenes.

1.1. MOTIVACIÓN Y OBJETIVOS

El mundo de la robótica hoy en día está sufriendo grandes progresos debido a su amplio uso en múltiples entornos. La robótica engloba muchas disciplinas como la mecánica, la electrónica, la informática, la inteligencia artificial, la ingeniería de control y la física. En los últimos años, se han realizado muchos avances en cada una de estas disciplinas, como en la inteligencia artificial. Esta área se encarga de diseñar entidades capaces de resolver problemas basándose en la inteligencia humana, lo cual puede aportar múltiples aplicaciones al mundo de la robótica. Dentro de este campo se puede encontrar entre muchos otros el de la visión artificial, ampliamente desarrollado y estudiado en la actualidad.

La visión artificial y la robótica están muy ligadas y en auge, debido a que dotar a un robot de una buena visión hace que aumente su autonomía permitiéndole entre otras cosas reconocer el entorno y ubicarse dentro de él. Uno de los problemas que surgen de este hecho es el conocido como el SLAM, Simultaneous Localization and Mapping. En este problema se trata de obtener un mapa del entorno a la vez que se ubica al robot en él. Para poner solución se emplean métodos como el scan matching, que trata de alinear imágenes diferentes pero con puntos en común. Esta alineación se puede llevar a cabo mediante algoritmos evolutivos, como el CLONALG, cuyas bases teóricas y funcionamiento se verán más adelante.

Los principales objetivos de este trabajo son dos. Por un lado es llegar realizar el scan matching utilizando el algoritmo CLONALG. El CLONALG es un algoritmo evolutivo cuyo funcionamiento se basa en principios presentes en el sistema inmunológico humano y que puede ser aplicado en múltiples problemas tanto de reconocimiento de patrones como de optimización. En este caso se debe aplicar a un problema de optimización que es la realización del matching entre dos imágenes. Partiendo del trabajo anterior [1], realizado por Fernando Martín Monar, entre otros, se debe alcanzar este fin. Para ello se parte de la utilización de las propiedades del color en imágenes y a partir de ahí el alumno debe ser capaz de implementar el algoritmo de forma que se obtengan las soluciones requeridas. Para ello partiendo de cero se debe hacer un estudio recopilando toda la información necesaria para después aplicarla y diseñar la solución.

El aprendizaje que se adquiere durante el estudio y desarrollo de todo este proceso es el otro objetivo de este trabajo. Al realizar un TFG (Trabajo Fin de Grado) se adquieren ciertas competencias que culminan la formación del alumno. Entre estas competencias está el saber enfrentarse a un proyecto de mayor envergadura de forma individual y que puede ser similar a los que hay en el mundo laboral, lo que conlleva realizar actividades de organización, documentación, diseño y estudio, entre otras.

1.2. HERRAMIENTAS EMPLEADAS

Para la realización del matching entre imágenes es necesario disponer de diferentes pares de imágenes con los que trabajar. Hoy en día en el mercado se han desarrollado sensores capaces de dar información de las imágenes tanto de distancia como de color. Uno de estos sensores es el Microsoft Kinect. Este sensor debido a su robustez, las buenas prestaciones que ofrece y su precio es, hoy en día, utilizado en múltiples proyectos científicos y tecnológicos. [3]

FIGURA 1.- SENSOR KINECK.

Toda la solución del presente trabajo se ha llevado a cabo mediante el software MATLAB, MATrix LABoratoy [4]. Este software es una herramienta ampliamente utilizada en ámbitos matemáticos debido a su alto potencial. Ofrece un entorno de desarrollo integrado y cuenta con su propio lenguaje de programación, el cual dispone de una extensa biblioteca de funciones que facilitan su uso. Debido a su gran demanda está disponible para múltiples plataformas.

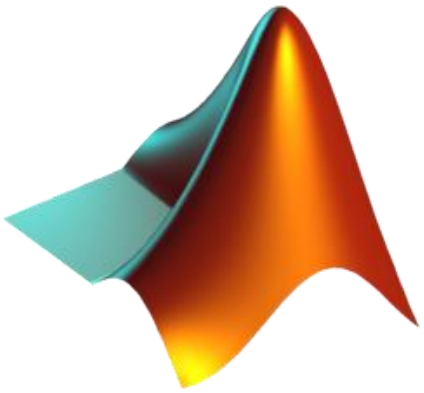

FIGURA 2.- MATLAB.

Cuenta con muchas prestaciones entre las que destacan el cálculo numérico, utilizando complejas funciones matemáticas; el análisis, visualización y modelado de datos, la programación y el desarrollo de algoritmos en lenguaje de alto nivel y el desarrollo y distribución de aplicaciones que permite desarrollar y compartir aplicaciones en forma de código, ejecutables o componentes de software.

1.3. ESTRUCTURA

El presente trabajo se ha estructurado de modo que inicialmente se haga una introducción teórica tanto del problema tratado como de las técnicas empleadas en la solución para, finalmente, mostrar los resultados experimentalmente obtenidos tras aplicar la solución desarrollada.

En primer lugar se hace un análisis del estado del arte, para pasar a los capítulos en los que se sintetizan las bases teóricas de todo el trabajo. Primero se describe el problema del scan matching y, a continuación, se explica el funcionamiento de los algoritmos evolutivos. Finalmente, se hace un desarrollo más específico del algoritmo implementado en este trabajo, el CLONALG.

Una vez asentadas las bases teóricas se pasa a explicar la solución implementada y mostrar los resultados experimentales obtenidos tras aplicarla. Se finaliza con una conclusión de todo el trabajo realizado y de los resultados obtenidos.

2. ESTADO DEL ARTE

Las dos técnicas empleadas en este trabajo tienen que ver con el scan matching relacionado directamente con el problema del SLAM, Simultaneous Localization and Mapping, y el uso de algoritmos evolutivos para resolver problemas de optimización. A continuación se explicará el estado del arte de estas dos técnicas.

2.1. MAPPING Y PROBLEMA DEL SLAM

En el mundo de los robots móviles aparece el denominado problema del SLAM, Simultaneous Localization and Mapping. Este problema trata de construir mapas de entornos inicialmente desconocidos y a la vez ubicarse dentro de ellos. Este problema fue introducido por Leonard and Durrant-White [5] basándose en el trabajo de Smith [6]. Debido al gran desarrollo que han sufrido los sensores 3D en los últimos años el interés por los mapas 3D se ha incrementado considerablemente.

La tarea para resolver este problema es compleja debido a varios factores como:

- Dificultad para representar entornos desconocidos.
- El ruido que aportan los sistemas sensoriales.
- Los errores cometidos en los modelos utilizados debido al uso de aproximaciones.

Para desarrollar soluciones óptimas se busca la manera de obtener datos sobre el entorno y procesarlos de forma que se consigan sólo aquellos que sean estrictamente prescindibles. Y obviar los que no arrojen información relevante para evitar trabajar con un exceso de información que además puede introducir errores innecesarios. La extracción de esta información del entorno es compleja debido a su carácter dinámico y a que los sensores no llegan a ser perfectos.

El procesamiento de los datos obtenidos del entorno también presenta complejidades. Se debe poder implementar soluciones que trabajen en tiempo en

real, y por lo tanto es fundamental acotar la cantidad de información de la que se dispone y determinar los recursos computacionales necesarios para la realización de mapas.

Uno de los principales cometidos del mapping es el scan matching, tarea en la que se basa principalmente este trabajo y que se explicará con mayor detenimiento en el capítulo 3. El scan matching consiste en realizar una estimación métrica tanto de posición como de orientación entre un par de imágenes. Esta estimación está muy ligada al problema del SLAM debido a que la información del entorno que va ser representado es habitualmente dada por sensores con información de distancia, es decir, dan información sobre la distancia a la que están los objetos. Esta aportación aplicada al scan matching ha abierto todo un mundo de posibilidades en la localización de robots móviles.

El Microsoft Kinect [3] es uno de estos nuevos sensores capaces de dar información tanto de distancia como de color. Es ampliamente conocido y utilizado tanto para usos de entretenimiento como para proyectos y desarrollos científicos y tecnológicos. La información que aporta de color puede ser utilizada para la creación de mapas volumétricos, lo cual hace que este sensor tenga diversas aplicaciones. Además esta información de color aporta ventajas que pueden ser empleadas a la hora de realizar el scan matching.

En este trabajo se va a desarrollar un proceso para realizar el matching entre pares de imágenes. Esta alineación se conseguirá mediante la implementación de un algoritmo evolutivo utilizando información de las imágenes procesada adecuadamente.

2.2. PROBLEMA DE OPTIMIZACIÓN

El problema de optimización [7] es un problema presente en el mundo de las matemáticas, estadística, ciencias de la computación y economía. Su objetivo es seleccionar atendiendo algún criterio previamente especificado al mejor elemento de un conjunto de elementos. Dada una función real y un conjunto de valores de entrada se calcula del valor de la función. Este conjunto de valores de entrada se debe elegir de forma que maximice o minimice la esta función. Por lo tanto la optimización da los mejores valores para una función de coste o función objetivo dentro de un dominio definido. Hay una amplia variedad tanto de funciones de coste como de dominios empleados.

Para resolver el problema de optimización se usan generalmente métodos iterativos que convergen en una solución, algoritmos cuya terminación es finita o métodos heurísticos que dan soluciones aproximadas aunque no converjan estrictamente. Dentro de las heurísticas se puede encontrar la siguiente variedad:

- Evolución diferencial.
- Algoritmos genéticos.
- Algoritmos de búsqueda diferencial.
- Relajación dinámica.
- Nelder-Mead.
- Optimización artificial de la colonia de abejas.
- Optimización por enjambre de partículas.
- Ascenso a montañas.

En este trabajo se dará solución a este problema de optimización desarrollando e implementado un algoritmo genético. Los algoritmos genéticos forman parte de los llamados algoritmos evolutivos y hacen progresar una población inicial utilizando acciones aleatorias como las empleadas en la evolución biológica. Se fundamentan en la selección natural y la supervivencia de los individuos más fuertes presentes en el mundo biológico. Un algoritmo evolutivo parte de una población inicial de individuos los cuales se modifican para quedarse con los mejores y rechazar los peores.

Se puede encontrar una amplia cantidad de algoritmos evolutivos. Entre esta variedad están el DE, y el CLONALG.

El DE es un algoritmo que emplea una búsqueda estocástica directa. Fue propuesto por Rainer Storn y Kenneth Price [8] [9] y surgió como una nueva heurística para lograr minimizar funciones no lineales y no diferenciales en el espacio continuo. Se consiguió crear un método que daba mejores resultados y de forma más rápida. Tienen múltiples ventajas como su robustez, el uso de menos parámetros de ajuste que otros algoritmos y su eficiencia a la hora de resolver problemas de optimización. Debido a estas ventajas este algoritmo ha sufrido un gran auge en los últimos años.

El CLONALG [2] es un algoritmo utilizado en los sistemas inmuno artificiales. Estos sistemas son un modelo computacional basado en el sistema inmunológico humano y tienen la capacidad de hacer tareas tan diversas como el aprendizaje, el reconocimiento de patrones, la adquisición de memoria, la generación de diversidad, la detección distribuida y la optimización.

3. SCAN MATCHING. PROPIEDADES DEL COLOR.

El scan matching se define como un método para calcular la relación espacial que hay entre dos imágenes próximas. Su uso y aplicación están fuertemente ligados a la localización de robots móviles debido a que la información del entorno suele venir dada por sensores que reciben datos del exterior. Esta es una de las aplicaciones más extendidas que tiene en el campo de la robótica, entre otras muchas.

Los grandes desarrollos llevados a cabo en el mundo de la visión artificial han hecho que el problema del SLAM, *Simultaneous Localization and Mapping*, [5] [6] sea uno de los asuntos más populares en el mundo de los robots móviles. Tanto la capacidad y eficiencia de los sensores 3D utilizados como el desarrollo de potentes algoritmos han hecho que se produzcan grandes progresos en este campo. Y así se han podido desarrollar algoritmos que son capaces de crear mapas de entornos desconocidos a la vez que se exploran, es decir, resolver el problema del SLAM.

A lo largo de la historia se han desarrollado varios métodos para realizar el *matching* o alineación entre imágenes implementando diferentes algoritmos y utilizando diferentes datos procedentes de la información que pueden dar las imágenes. Se han propuesto varias soluciones como por ejemplo una técnica basada en el alineamiento de imágenes 3D utilizando el ICP [10], *Iterative Closest Points*, en combinación con métodos heurísticos. También son muy comunes los algoritmos para la reconstrucción de imágenes de entornos interiores y exteriores con robots móviles [11]. Así mismo se han desarrollado y construido mapas MLS, *Multi-Level Surface* [12] o se han diseñado técnicas de alineamiento basadas en transformaciones de distribuciones normales NDT, *Normal Distributions Transform*. [13]. Debido a todo este progreso y desarrollo se ha demostrado que las técnicas tradicionales para resolver el problema del SLAM dan peores resultados cuando este se extiende a las seis dimensiones, 6D, tres lineales y tres angulares.

Para realizar el scan matching hay que valorar diferentes parámetros con los que se puede trabajar con las imágenes. Por un lado los mapas utilizados para el scan matching puede ser de dos dimensiones, 2D o de tres dimensiones, 3D. A partir de estos últimos, con la información obtenida de las tres dimensiones lineales se puede llegar a trabajar en seis dimensiones, 6D, utilizando tres para las

coordenadas lineales del mapa y otras tres para las angulares. Por otro lado, los métodos empleados pueden ser tanto locales como globales. En los métodos locales se alinean imágenes de forma individual, mientras que en los métodos globales se trabaja con un modelo global y una imagen actual.

También a la hora de resolver el problema del scan-matching se puede distinguir entre tres tipos de metodologías de tratamiento de imágenes, dependiendo del uso de características previas a la extracción de puntos o no. Así se puede emplear un método en el que no se requiera de ninguna información o características previas al procesamiento, es decir, se utilicen únicamente los puntos dados por las imágenes. O, en el caso contrario, se puede disponer de estructuras distinguibles del entorno previas al procesamiento. O, como última opción, se puede dar una mezcla de los dos métodos anteriores, implementando un procedimiento que busque correspondencia entre puntos y características de las imágenes.

Para iniciar el matching entre imágenes se requiere la adquisición de, al menos, dos imágenes del entorno detectadas por el robot de las que se saca información de las coordenadas espaciales para procesarla adecuadamente con la implementación de un algoritmo. Con este proceso se llega a determinar la transformada que hace que los puntos de interés de las dos imágenes se alineen. Una vez alineados, con la información obtenida al realizar el proceso es posible calcular la variación espacial que hay entre estas dos imágenes.

De las imágenes que se utilizan hay que obtener los putos de interés con los que se trabajará en el algoritmo. Estos puntos se pueden tomar siguiendo diferentes técnicas. A lo largo de la historia se han desarrollado y estudiado diferentes métodos y se observado que el mayor problema que se presenta a la hora de resolver el problema del scan matching es cuando hay cambios considerables de la orientación entre imágenes. Una de las técnicas que minimizan este problema es utilizar propiedades del color de las imágenes para sacar los puntos con los que trabajar, ya que hay una serie de propiedades del color que se ven menos afectadas por la rotación en imágenes 3D.

Gracias a los últimos progresos en los sensores utilizados en la percepción del entorno podemos utilizar la información procedente del color de imágenes en múltiples aplicaciones. El ámbito de la robótica ha sido muy influenciado por estos progresos, y más concretamente el campo del *mapping*. Esto es debido a que el uso de las propiedades del color tiene varias ventajas, como que es compatible con información sobre la distancia de los objetos de la imagen y su bajo coste. Por ello

es posible incorporar propiedades de color en los pasos seguidos en el scan matching.

Por lo tanto, una vez obtenidas las imágenes del entorno es necesario extraer puntos de éstas para que nos den información sobre la posición y orientación y así poder compararlos entre una imagen y otra. Y estos puntos de interés se pueden sacar utilizando propiedades del color, como la extracción de transiciones de color entre diferentes conjuntos de puntos de la imagen.

Con esta técnica se obtienen los puntos con los cambios más significativos de color y que pueden ser empleados como puntos de interés a la hora de realizar el matching. Para desarrollar este método adecuadamente se utiliza como unidad de medida de la diferencia de color el Delta E y, además, los datos se incluyen en la función de coste utilizada posteriormente en el algoritmo evolutivo que realizará el scan matching.

Delta E es una medida de la diferencia de color utilizada por la CIE, International Commission on Illumination. Esta medida emplea la distancia euclídea en espacios de color independiente. A lo largo de la historia se han desarrollado diferentes métodos de cálculo, como el CIE76, CIE94 o CIEDE2000.

Los sensores actuales nos dan tanto información de la distancia en la imagen como del color en RGB, lo que se conoce como información RGB-D. En el espacio de color RGB, cada color es representado por tres componentes independientes que se corresponden con el rojo (R), el verde (G) y el azul (B).

$$\text{Colour} = (\text{R}, \text{G}, \text{B}) \qquad (1)$$

Estos componentes tienen un valor cada uno de ellos que oscila entre cero y un máximo, así, por ejemplo el blanco se obtiene cuando los tres alcanzan el valor máximo, y el negro cuando alcanzan el mínimo. El resto de la gama de colores se adquieren variando cada componente en este rango. La simplicidad de este modelo de color ha sido de gran utilidad en diferentes campos tecnológicos, como en la electrónica o en la fotografía. Pero también presenta ciertos inconvenientes, como que los valores RGB pueden diferir de un dispositivo a otro para el mismo objeto, o que las condiciones de iluminación pueden producir cambios notables. Por lo tanto, el espacio de color RGB no da un color absoluto para un mismo objeto.

Utilizando el espacio de color RGB es posible implementar un método para calcular la diferencia de color entre puntos cercanos de una imagen, utilizando la siguiente ecuación

$$d_c = \sqrt{\Delta R^2 + \Delta G^2 + \Delta B^2} \qquad (2)$$

donde ΔR es la diferencia de rojo entre puntos cercanos, ΔG la diferencia de verde, ΔB la diferencia de azul y d_c es la diferencia de color global. Una vez obtenida la diferencia de color entre puntos comparándola con un valor de referencia se pueden obtener los cambios de color más importantes en conjuntos de puntos. Y ya conocidos los puntos significantes se puede proceder a iniciar el algoritmo para realizar el matching entre un par de imágenes.

Delta E () es la medida de la diferencia de color propuesta por el CIE, Comisión Internacional de la Iluminación o Commission Internationale de l´Éclairage, donde delta representa la variación y E es la sigla de la palabra alemana Empfindung que significa percepción. El objetivo de esta variable es medir las diferencias de color que son perceptibles por el ser humano, estableciendo un valor denominado JND, Just Noticeable Difference.

A partir del espacio de color RGB el CIE propuso en 1931 el espacio CIE XYZ que derivó en el espacio de color Lab [14] [15]. Este espacio de color está formado por tres componentes, L, a y b. El primer parámetro, L, representa la luminosidad del color, un valor elevado indica blanco y uno bajo negro. Los dos siguientes parámetros representan la posición entre rojo y verde, a, y la posición entre amarillo y azul.

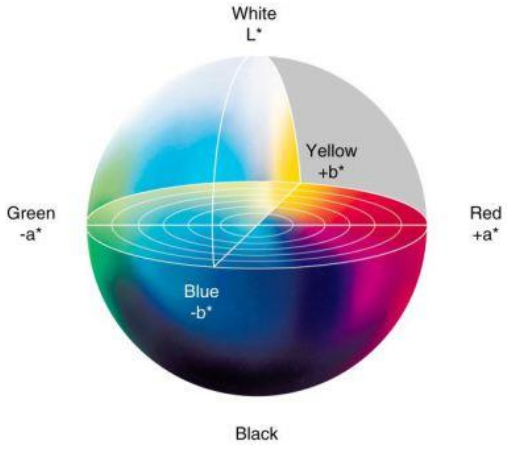

FIGURA 3.- ESPACIO DE COLOR CIELAB.

Estas coordenadas evolucionaron y se originó el espacio de color CIELAB donde las tres componentes pasaron a ser L*a*b*. Los asteriscos denotan cierto cambio a la hora de obtener el valor de cada parámetro para emular de forma más parecida la no linealidad del ojo humano. De esta forma los cambios de color uniformes que se produzcan entre dos puntos se pueden aproximar por la distancia euclídea entre ellos.

A lo largo de los años utilizando la variable delta E se han sacado diferentes versiones intentado mejorar el problema de la no uniformidad del espacio CIELAB debido a que el ojo humano no es igual de sensible para unos colores que para otros. Así se han obtenido las versiones CIE76, CIE94 y CIEDE2000 entre otras.

Una vez detectadas las diferencias de color notables se conseguirán los puntos necesarios para utilizarlos en el proceso que se requieran.

4. ALGORITMOS EVOLUTIVOS

La inteligencia artificial, IA, se define como el área encargada de diseñar y crear entidades capaces de resolver problemas por sí mismas basándose en la inteligencia humana, y empleando para ello las ciencias de la computación, la lógica y la filosofía. Una de las ramas que conforman la inteligencia artificial es la computación evolutiva, donde se engloban problemas de optimización combinatoria y se inspira en los mecanismos de la evolución biológica. Dentro de la computación evolutiva se pueden distinguir tres corrientes diferentes de investigación, la programación evolutiva, las estrategias evolutivas y los algoritmos genéticos, estos últimos enmarcados dentro de los algoritmos evolutivos. Estas corrientes fueron impulsadas durante los años 60 y 70 por varios científicos como Lawrence J.Fogel, Ingo Rechenberg y John H. Holland, entre otros. [16] [17]

Los algoritmos evolutivos se utilizan para dar solución a todo tipo de problemas debido a que no se necesita ninguna especificación del entorno, lo que ha llevado a que se puedan aplicar en campos tan variados como la ingeniería, la robótica, la física, la química, la biología, la economía, el marketing o el arte.

La complejidad computacional de estos algoritmos ha hecho que en muchas ocasiones se evite su aplicación. Esta complejidad procede de la evaluación de la función de coste con la que trabajan. Esta función es la que un algoritmo evolutivo debe optimizar de forma que la solución dada alcance máximo o mínimos, dependiendo del tipo de problema para el que haya implementado. Otro de los problemas a la hora de aplicar este tipo de algoritmos es la codificación del entorno que debe realizarse para adaptarlo a los parámetros utilizados. Dependiendo del tipo de problema propuesto esta codificación puede ser más sencilla o compleja, e incluso puede llegarse a dar el caso en que no permita la aplicación del algoritmo.

Un algoritmo es un conjunto de instrucciones bien definidas, ordenadas y finitas que describe el proceso que se debe seguir para llegar a la solución de un problema específico. A diferencia de otros tipos de algoritmos, los algoritmos evolutivos dan soluciones basadas en postulados de la evolución biológica y toma como base las ideas de la evolución propuestas por Charles Darwin. Algunos de los mecanismos biológicos en los que se inspira un algoritmo evolutivo son la reproducción, la recombinación, la selección y la mutación. El procedimiento que siguen aunque a priori parece simple puede involucrar complejas operaciones. Este procedimiento consiste en hacer evolucionar una población mediante la aplicación

de una serie de operadores repetidas veces hasta lograr la solución adecuada u óptima. La población está compuesta por un conjunto de posibles candidatos, individuos, de los cuales, mediante los valores dados por una función de coste, se determina la calidad de las soluciones que dan.

El procedimiento básico que sigue un algoritmo evolutivo es el mostrado en la Figura 4. Donde P es una población inicial que representa los datos del problema del que se parte. Esta población está compuesta por diferentes individuos candidatos todos ellos a formar parte de la solución final. La función de coste o función objetivo es la que el algoritmo debe optimizar. Cada individuo de la población P tiene un valor de coste y mediante el análisis de este valor se determina cuando un candidato es apto y cuando no para formar parte de la solución final. Esta operación se hace en el siguiente paso del algoritmo y es la denominada selección. De este modo se consigue obtener una población PI de candidatos más propensos a llegar a la solución. Finalmente se aplican una serie de mecanismos de variación, como la clonación, mutación o cruce, entre otros muchos. Una vez modificada la población mediante estos mecanismos se vuele a iniciar todo el proceso, hasta que se cumpla la condición de parada, que puede ser llegar a un determinado valor o realizar un número concreto de iteraciones.

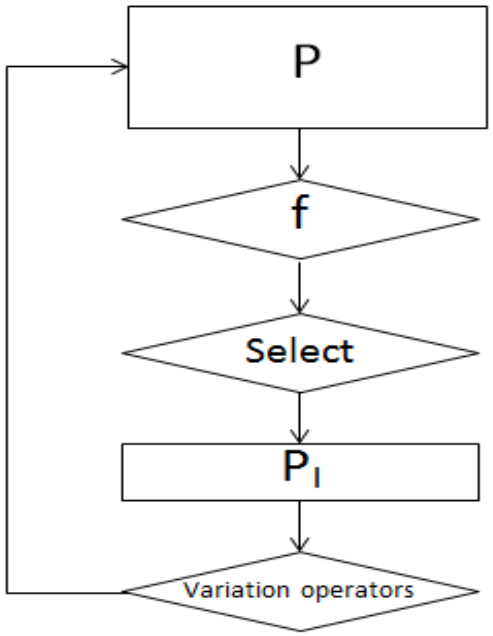

FIGURA 4.- DIAGRAMA BLOQUES ALGORITMO EVOLUTIVO.

Todos estos procesos son comunes para todos los algoritmos evolutivos pero cada algoritmo tiene sus peculiaridades.

Los principales tipos de algoritmos evolutivos que nos podemos encontrar son los siguientes:

- Algoritmos genéticos.
- Programación genética.
- Programación evolutiva (EA).
- Expresión genética de programación (GEP).
- Estrategia evolutiva.
- Estrategia diferencial.
- Neuroevolución.
- Sistemas clasificadores de aprendizaje (LCS).

Todos siguen el mismo patrón pero cada tipo presenta unas particularidades que lo hacen más útil para resolver unos problemas u otros. De hecho dentro de cada tipo de algoritmos evolutivos se han desarrollado a su vez diferentes algoritmos con la misma base de funcionamiento pero diferentes implementaciones. Por lo tanto, debido a la amplia gama de algoritmos evolutivos presentes hoy en día es necesario hacer un estudio detallado del problema para el que se quiere aplicar e intentar buscar el algoritmo que mejor se asocia a él, ya que, como ya se ha mencionado, la complejidad computacional puede llevar a que no se alcance una solución del problema o el proceso sea poco eficiente.

Dentro de toda la variedad de los algoritmos evolutivos en este trabajo habría que hacer mención especial al Differential Evolution [8] y al CLONALG [2]. El Differential Evolution (DE), es un algoritmo evolutivo que se basa en un método de búsquedas estocásticas. Es un algoritmo ampliamente utilizado para diferentes propósitos debido a sus buenos resultados. El algoritmo parte de una población inicial en al que cada individuo presenta una valor para una función de coste. La idea es optimizar el valor de esta función de coste y para ello utiliza los mecanismos de mutación, recombinación, selección y descarte, en este orden. El algoritmo detiene su funcionamiento cuando se alcanza un valor de convergencia dado.

Este algoritmo fue utilizado en el proyecto RGB-D DE-based Scan Matching [1] que tiene el mismo objetivo que este trabajo, resolver el problema de scan matching.

Otro importante algoritmo evolutivo es el CLONALG que presenta una serie de diferencias en su funcionamiento. Este algoritmo está basado en la teoría de selección clonal presente en el sistema inmunológico humano. Al igual que en el algoritmo DE se parte de una población inicial formada por una serie de individuos que son posibles candidatos. A continuación también se requiere una función de

coste cuyo valor se va a optimizar. La diferencia radica en los operadores de variación que se emplean. El CLONALG realiza inicialmente una selección para seguir con la clonación de estos elementos seleccionados. Seguidamente los muta y finalmente vuelve a seleccionar los mejores candidatos para formar parte del conjunto final.

El CLONALG es el algoritmo evolutivo en el que se basa este trabajo, y más adelante se explicará con mucho más detalle su funcionamiento, que, aunque a priori puede parecer simple si no se estudia detenidamente puede dar problemas a la hora de alcanzar la convergencia.

5. CLONALG

5.1. INTRODUCCIÓN

El CLONALG [2] es un algoritmo basado en la teoría de la selección clonal que se sustenta en el funcionamiento del sistema inmunológico adaptativo. Mediante esta teoría se va desarrollando el algoritmo para dar respuesta a dos tipos de problemas computacionales que son el problema de reconocimiento de patrones y el problema de optimización.

Debido a su funcionamiento y propiedades puede ser aplicado del mismo modo que un algoritmo evolutivo. El CLONALG es un algoritmo en el que una población con potenciales candidatos evoluciona hasta que uno de los miembros se convierte en la mejor solución. Al igual que en los algoritmos evolutivos utiliza operadores de selección y variación, como la hipermutación, que guían a la población hasta conseguir la solución óptima.

Hay una serie de características interesantes que presenta el sistema inmunológico humano y que pueden ser útiles a la hora de desarrollar algoritmos basados en su funcionamiento y que den respuesta ciertos problemas de una forma eficaz en determinadas áreas de investigación. La inmunidad es una capacidad defensiva de un organismo frente a agentes patógenos. Está constituida por un complejo sistema de biomoléculas y células capaces de neutralizar e incluso destruir a los agentes infecciosos. Por lo tanto, una de sus primeras funciones será distinguir los agentes no propios de los propios y mediante un complejo proceso de aprendizaje y memoria lograr neutralizar a los agentes externos.

La respuesta inmunitaria puede ser de dos tipos, innata o adaptativa. El CLONALG está basado en esta última. La respuesta inmunitaria innata actúa con carácter general contra agentes patógenos o infecciosos y su poder de acción no mejora con sucesivas infecciones, es decir, se mantiene invariante, lo cual no resulta interesante a la hora de desarrollar algoritmos que vayan evolucionando hasta llegar a una solución óptima. Pero, la respuesta adaptativa sí que tiene una resistencia que mejora tras sucesivas infecciones utilizando para ello una memoria inmunológica. Este mecanismo de aprendizaje hace que tenga una acción muy específica contra agentes infecciosos. Por ello resulta tan interesante a la hora de

aplicarlo en .sistemas adaptativos capaces de actuar en un amplio rango de tareas en diferentes áreas de investigación, ya que nos permite ir obteniendo mejores resultados tras sucesivas series (infecciones) hasta llegar al resultado óptimo (ausencia de enfermedad).

Además el principio en el que se basa el CLONALG, la teoría de la selección clonal, se puede considerar como un proceso evolutivo debido a ciertas características que presenta. Charles Darwin postuló en su teoría de la evolución los tres grandes principios que cumple un proceso evolutivo y son la diversidad, la variación genética y la selección natural. Estos tres principios aparecen en la teoría de la selección clonal pudiéndose considerar por tanto como un proceso evolutivo más.

5.2. PRINCIPIO DE SELECCIÓN CLONAL

El sistema inmunológico adaptativo ha ido evolucionando hasta llegar a actuar como lo hace hoy en día, dando una respuesta inmunitaria mayor. Esto es debido en gran medida a la llamada memoria inmunológica. Las células que aparecen en el sistema inmunitario adaptativo son los linfocitos, un tipo especial de leucocitos, y en concreto aparecen dos tipos de células principalmente, las células B y T, que son las encargadas de llevar a cabo este complejo trabajo.

Los procesos más destacados del principio de selección clonal [18] [19] son los siguientes. En primer lugar la proliferación y diferenciación de las células que han sido capaces de reconocer al antígeno. A continuación, aparece un proceso de maduración, mutación somática, en diversos patrones de los anticuerpos, donde se generan nuevos cambios genéticos aleatorios que son los encargados de lidiar contra el antígeno reconocido. Y, finalmente, aparece un proceso de selección de células B, donde, aquellas células con mayor afinidad respecto al antígeno entraran en un conjunto de células de memoria, y las que presenten menor afinidad serán eficientemente eliminadas o modificadas, en su defecto.

En el transcurso de vida de un organismo, éste se puede encontrar repetidamente con un determinado antígeno. En la respuesta inmune podemos distinguir tres tipos de respuestas: la respuesta primaria, la secundaria y la reactividad inmunológica cruzada, las cuales ocurren siguiendo una línea temporal. En primer lugar aparecerá la respuesta primaria, en segundo lugar la secundaria y finalmente se habla de reactividad inmunológica cruzada para todas las demás respuestas que se den en las sucesivas apariciones del antígeno.

En la primera exposición el antígeno se encontrará con un bajo número de células B con baja afinidad pero éstas producirán un tipo de anticuerpo con una afinidad diferente ligada al antígeno. Por lo tanto, el primer paso que realiza el sistema inmunológico adaptativo es reconocer el antígeno o patógeno infeccioso que no es propio del organismo. Cualquier molécula no propia del organismo puede ser reconocida como un antígeno por el sistema inmunológico adaptativo. Este procedimiento es llevado a cabo por las células T que son capaces de reconocer un objetivo no propio del organismo, como un patógeno.

A continuación, cada patógeno es recordado por un antígeno característico que desencadena una formación de anticuerpos que son capaces de lidiar con este único antígeno, como se muestra en la figura 2. Las células B son las encargas de generar los antígenos. Cada célula B contiene al receptor específico de cada antígeno, es decir, las moléculas que forman los anticuerpos. Estos linfocitos B logran identificar los patógenos cuando los anticuerpos de su superficie se unen a determinados antígenos no propios.

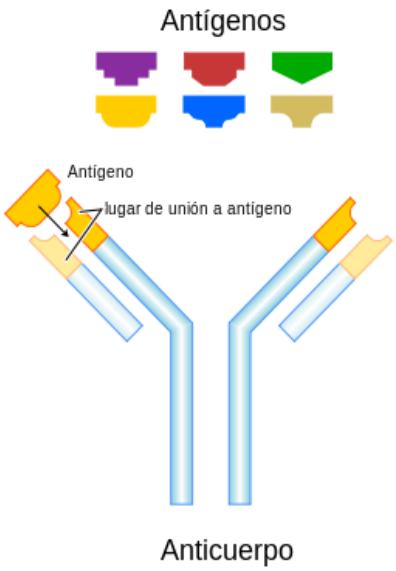

FIGURA 5.- IDENTIFICACIÓN ANTÍGENOS POR ANTICUERPOS.

Una vez ligado el anticuerpo al antígeno y activado el procedimiento de ataque contra el agente patógeno las células B y T se dividen y se crean millones de copias de ellas, en lo que se denomina respuesta primaria. De todas estas copias algunas pasan a formar parte de un conjunto de células de memoria con un largo periodo de vida. Estas células son las encargadas de recordar cada patógeno específico que se hayan encontrado a lo largo de la vida de un ser humano. Todo este proceso queda reflejado en la figura que se muestra a continuación.

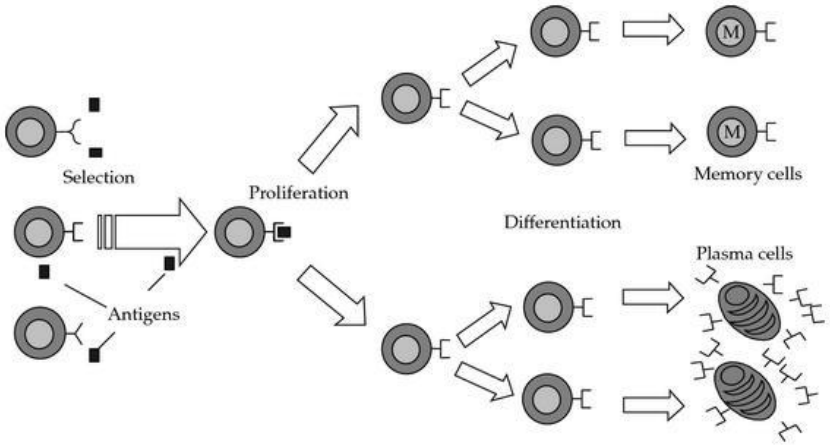

FIGURA 6.- PRINCIPIO DE SELECCIÓN CLONAL.

Finalmente, gracias a estas células de memoria se consiguen establecer estas respuestas específicas en sucesivas infecciones para poder eliminar lo más rápidamente posible al patógeno infeccioso. La efectividad de la respuesta inmunológica, por lo tanto, dependerá en gran medida de estas células de memoria asociadas desde la primera infección y que son capaces de producir anticuerpos de alta afinidad a lo largo de sucesivos encuentros con el antígeno correspondiente. Es decir, la respuesta inmune, en lugar de empezar desde cero en cada infección, con esta estrategia se asegura una mejora de la efectividad y velocidad en la respuesta, y, por lo tanto, podemos hablar de un mecanismo de aprendizaje inmune adaptativo.

Después de la respuesta primaria, aparecerá lo que se denomina respuesta secundaria. Ésta se caracteriza por un periodo de respuesta más corto, una velocidad en todo el proceso mayor y una alta concentración de anticuerpos con elevada afinidad.

Tras sucesivos encuentros, el sistema inmunológico se va a ir adaptando a lo largo del tiempo de vida a las infecciones atacantes y se preparará para futuros ataques de antígenos similares, en lo que se denomina reactividad inmunológica cruzada.

El proceso de vacunación se basa en este mecanismo de aprendizaje, el cual también se ha llevado a otros campos de aplicación como en la computación y su uso en las redes neuronales, perteneciente al ámbito de la inteligencia artificial. Algunas características de esta memoria asociativa son especialmente interesantes a la hora de usar este tipo de mecanismo en los sistemas artificiales inmunes. Una de ellas es que el patrón almacenado inicialmente se recupera en sucesivas versiones. Y la otra característica importante es que son sistemas muy robustos, no aparece ruido en los datos ni fallos en los componentes de memoria.

Po lo tanto, resumiendo, la memoria y el aprendizaje inmune es obtenido siguiendo los siguientes pasos:

1) Exposición repetida ante un antígeno.
2) Maduración de la afinidad de las moléculas receptoras.
3) Disminución del grado de infección crónica.
4) Reactividad inmunológica cruzada.

De todos estos pasos hay uno en el que merece la pena detenerse y analizarlo más detalladamente debido al gran papel que juega en el principio de selección clonal, y es el proceso de maduración de la afinidad donde se lleva a cabo la hipermutación y la selección de los anticuerpos con mayor afinidad.

Lo que se conoce como maduración de la respuesta inmune es debido a que los anticuerpos presentes en las sucesivas respuestas por haber pertenecido al conjunto de memoria tienen, en general, una mayor afinidad que aquellos que participaron en la respuesta primaria. Esta afinidad va aumentando debido a unos cambios aleatorios que se van introduciendo en los genes encargados de las interacciones entre los anticuerpos y el antígeno. Luego, una vez realizados estos cambios pasarán a formar parte del set de memoria los elementos con mayor afinidad.

Para lograr una rápida maduración de la respuesta inmune se necesitan continuas mutaciones de la población, pero la mayoría de estos cambios pueden llevar a empeorar los anticuerpos y hacerlos inservibles. Por lo tanto hay que controlar esta mutación, ya que podría darse el caso de que un anticuerpo útil mutara y pasara a tener una peor afinidad, lo cual nos conduciría a una mala solución. Para solucionar este problema el mecanismo de mutación debe tener en cuenta la afinad del anticuerpo, y así mutar de forma proporcional a esta. Así en células con una alta afinidad la hipermutación debe mantenerse inactiva o prácticamente inactiva. Y, en cambio, en células con una baja afinidad la hipermutación debe ser más agresiva y veloz. De este modo nos aseguraremos de que el mecanismo converja y llegue a una solución óptima.

5.3. ASPECTOS COMPUTACIONALES. EL ALGORITMO.

El uso del CLONALG [2] se ha utilizado, generalmente, para solucionar dos tipos de problemas, el reconocimiento de patrones y el problema de optimización. Aunque a la hora de implementar el algoritmo en ambos casos la base en la que se sustenta es la misma, hay alguna diferencia en su desarrollo.

La base del CLONALG se fundamenta en los principales aspectos de la teoría de la selección clonal y son comunes para las dos soluciones. Estos principios son los siguientes:

1. Conservación de un conjunto de memoria.
2. Selección y clonación de los anticuerpos con mayor afinidad.
3. Destrucción de los anticuerpos menos afines.
4. Mutación
5. Re-selección de los clones más afines y generación y mantenimiento de la diversidad.

El algoritmo se desarrolla trasladando estas bases teóricas a unas más matemáticas para poder utilizarlo en diferentes campos y aplicaciones. Así tendremos conjuntos de datos que representaran a los antígenos y anticuerpos presentes en el proceso inmunológico, y una serie de funciones que harán que estos parámetros se modifiquen hasta llegar a alcanzar la solución óptima.

RECONOCIMIENTO DE PATRONES

En el campo del reconocimiento de patrones el propósito principal es obtener información de objetos que permita establecer alguna relación entre ellos. Para ello primero se debe adquirir los datos, luego extraer las características más relevantes y, finalmente, trabajar con estas características para tomar decisiones y actuar al respecto. En este último paso es donde se puede aplicar el CLONALG.

Para poder realizar el reconocimiento de patrones vamos se utilizan dos poblaciones en el algoritmo, que no son otras que la de antígenos (Ag) y la de anticuerpos (Ab).

El algoritmo CLONALG para este uso sigue los siguientes pasos:

1. Del grupo inicial de antígenos, *Ag*, cuyo tamaño es *M*, se elige aleatoriamente un elemento, Ag_j, para ser presentado al conjunto de anticuerpos, *Ab*, con tamaño *N*.
2. Se calcula la afinidad que tienen todos los anticuerpos presentes con este antígeno seleccionado. Esta afinidad se guarda en un vector de afinidades *f*.
3. Los elementos que presenten mayor afinidad son seleccionados, formando un grupo de *n* anticuerpos, Ab_n.
4. Se clonan los elementos pertenecientes al grupo de los anticuerpos de alta afinidad, Ab_n. Esta clonación debe realizarse de forma proporcional a su afinidad, es decir, los elementos con mayor afinidad generarán más clones que los de menor afinidad. Así se formará el conjunto de clones *C*.
5. Los elementos clonados se mutan para aumentar la diversidad. El proceso de mutación debe ser inversamente proporcional a la afinidad, es decir, los elementos con mayor afinidad deben mutar menos, y los elementos con peor afinidad mutan más para que la población llegue a converger. *C** es el conjunto de clones mutados.
6. Con los nuevos elementos se calcula de nuevo la afinidad, *f** respecto al antígeno seleccionado inicialmente.
7. De este último conjunto de anticuerpos se selecciona el que tenga una mayor afinidad y pasa a ser un candidato para entrar a un conjunto de elementos de memoria, Ab_m. Si el nuevo candidato posee una mayor afinidad que el anticuerpo de memoria ya almacenado, entonces lo reemplazará y pasará a ser el anticuerpo de memoria.
8. El resto de anticuerpos con baja afinidad se reemplazarán por nuevos candidatos, conjunto Ab_d.

Todos estos pasos quedan resumidos en la siguiente figura:

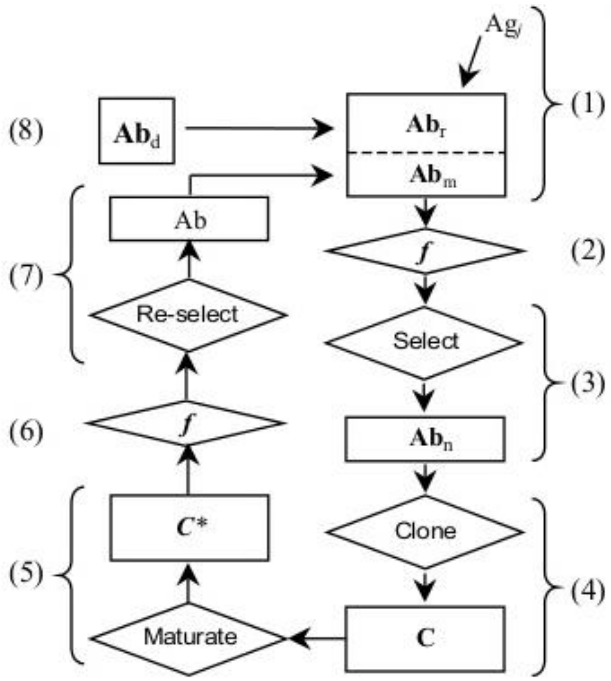

FIGURA 7.- DIAGRAMA BLOQUES CLONALG RECONOCIMIENTO DE PATRONES.

Mediante este algoritmo se logra mejorar la capacidad de aprendizaje gracias a cambios aleatorios y a la selección que hacen que la afinidad de los individuos vaya aumentando con cada iteración.

Como se puede observar hay una cantidad de parámetros y funciones que se pueden modificar de tal forma que el algoritmo se haga más eficiente, como son el tamaño del grupo de los antígenos, el tamaño del grupo de los anticuerpos, la función de afinidad, el número de clones a obtener y la forma proporcional de obtenerlos, el método de mutación o el número de iteraciones que se realizan. Para conseguir desarrollar el algoritmo y que sea lo más óptimo posible debe profundizarse en cada uno de los parámetros y estudiarlo detenidamente porque variará su forma de utilización dependiendo de la aplicación para la que se esté implementando.

El pseudocódigo del CLONALG para el reconocimiento de patrones se muestra en la siguiente figura:

```
// Pattern recognition pseudocode
for t = 1 to iter_max,
        for j = 1 to M,                                          % Step 1
                f(j,:)    := affinity(Ab, Ag(j,:));              % Step 2
                Abn(j,:)  := select(Ab, f(j,:), n);              % Step 3
                C(j,:)    := clone(Abn, β, f(j,:));              % Step 4
                C*(j,:)   := hypermut(C, f(j,:));                % Step 5
                f*(j,:)   := affinity(C*(j,:), Ag(j,:));         % Step 6
                Ab*       := select(C*, f*(j,:), 1);             % Step 7
                if f(Ab*) > f(Abm(j,:))
                        Abm(j,:)  := insert(Abm(j,:), Ab*);
                end
                Abd       := generate(d, L);
                Abr       := replace(Abr, Abd, f);               % Step 8
        end
end
```

FIGURA 8.- PSEUDOCÓDIGO CLONALG RECONOCIMIENTO DE PATRONES.

Donde,

Iter_max: número de iteraciones.
Ag: conjunto de antígenos (MxL).
M: tamaño del conjunto de antígenos.
L: longitud del conjunto de antígenos.
Ab: conjunto de anticuerpos (NxL).
F: función de afinidad.
N: tamaño de conjunto de clones generados a partir de la afinidad.
C: conjunto de clones.
B: factor de multiplicación utilizado en el cálculo del número de clones.
C*: conjunto de clones mutados.
F*: función de afinidad de los elementos clonados.
Ab*: conjunto de anticuerpos clonados y seleccionados.
Abm: conjunto de anticuerpos pertenecientes al set de memoria.
Abd: conjunto de nuevos anticuerpos que reemplazan a los de baja afinidad.
D: tamaño del conjunto Abd.
Abr: anticuerpos restantes.

OPTIMIZACIÓN

En los problemas de optimización se trata de buscar una solución que represente el valor óptimo para una función objetivo. El CLONALG debido a su funcionamiento es un algoritmo que se puede utilizar para esta búsqueda ya que partiendo de una población muta y selecciona aquellos elementos con mayor afinidad mejorando la progenie. El algoritmo realiza una búsqueda meticulosa, ya que se mejoran los elementos de forma local, y con los elementos nuevos que se van introduciendo se amplía el espacio de búsqueda.

Respecto al funcionamiento del CLONALG aplicado para el reconocimiento de patrones, visto anteriormente, se realizan una serie de modificaciones.

- Ya no es necesario tener una población de antígenos si no una función objetivo que será la que se optimizará buscando su máximo o mínimo. Por lo tanto lo nombrado anteriormente como afinidad, que era la que había entre un anticuerpo y el antígeno seleccionado, ahora pasa a ser un correspondiente valor de la función objetivo.

- Los anticuerpos que pasan a formar parte del set de memoria ahora pueden ser todo el conjunto de aquellos seleccionados en lugar de seleccionar sólo el anticuerpo que tenga la mejor afinidad.

- Se pueden seleccionar todos los elementos para clonarlos, en lugar de sólo los *n* mejores. Aunque para mantener el mejor anticuerpo generado en cada evolución se puede mantener uno o alguno de ellos, y así asegurar que la búsqueda no empeore.

- El número de clones generados para cada anticuerpo puede ser el mismo, y no tiene por qué ser proporcional a su afinidad como ocurría en el reconocimiento de patrones.

Con estas posibles modificaciones el diagrama de bloques del algoritmo queda así:

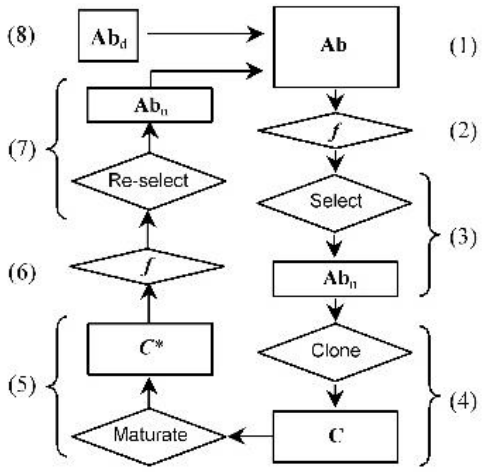

FIGURA 9.- DIAGRAMA BLOQUES CLONALG OPTIMIZACIÓN.

Los pasos que sigue ahora el algoritmo son los siguientes:

1. Conjunto aleatorio de anticuerpos, *Ab*, con tamaño *N*.
2. Cálculo del valor de la función objetivo para todos los anticuerpos presentes Este valor se guarda en el vector *f*.
3. Los elementos que presenten mayor afinidad son seleccionados, formando un grupo de *n* anticuerpos, Ab_n.
4. Se clonan los elementos pertenecientes al grupo de los anticuerpos de alta afinidad, Ab_n. Esta clonación puede realizarse de forma proporcional a su afinidad, es decir, los elementos con mayor afinidad generarán más clones que los de menor afinidad. O se puede clonar a todos los anticuerpos en la misma medida. Se formará el conjunto de clones *C*.
5. Los elementos clonados se mutan para aumentar la diversidad. El proceso de mutación debe ser inversamente proporcional a la afinidad, es decir, los elementos con mayor afinidad deben mutar menos, y los elementos con peor afinidad mutan más para que la población llegue a converger. *C** es el conjunto de clones mutados.
6. Con los elementos mutados se calcula de nuevo el valor de la función objetivo, *f**.
7. Se selecciona todo el conjunto de los anticuerpos que optimizan la función objetivo para formar parte del set de memoria. También se puede seleccionar el elemento que tenga una mayor afinidad para ser un candidato a entrar en el

conjunto de elementos de memoria, Ab_m , en lugar de seleccionar a todo el conjunto.

8. El resto de anticuerpos con baja afinidad se reemplazarán por nuevos candidatos, conjunto Ab_d.

El pseudocódigo para la versión de optimización se muestra en la siguiente imagen:

```
// Optimization pseudocode
for t = 1 to iter_max
        f    := decode(Ab);          % Step 2
        Abn  := select(Ab, f, n);    % Step 3
        C    := clone(Abn, β, f);    % Step 4
        C*   := hypermut(C, f);      % Step 5
        f*   := decode(C*);          % Step 6
        Abn  := select(C*, f*, n);   % Step 7
        Ab   := insert(Ab, Abn);
        Abd  := generate(d, L);
        Ab   := replace(Ab, Abd, f); % Step 8
end
f := decode(Ab);
```

FIGURA 10.- PSEUDOCÓDIGO CLONALG OPTIMIZACIÓN.

Por lo tanto, dependiendo del tipo de problema al que nos enfrentemos el funcionamiento del algoritmo puede verse levemente modificado siempre tratando de llegar a la solución correcta.

6. DISEÑO DE LA SOLUCIÓN

6.1. PLANTEAMIENTO DEL PROBLEMA

Una vez realizado el estudio teórico sobre el scan matching, los algoritmos evolutivos y el CLONALG se va a proceder a explicar paso a paso el diseño que se ha desarrollado para buscar la mejor solución al problema planteado

El objetivo de este trabajo es realizar el scan matching entre dos imágenes utilizando el algoritmo evolutivo CLONALG. Para realizar esta alineación hay que buscar una solución a partir de los datos de los que se dispone. Partiendo de que se quiere obtener la rotación y traslación necesarias para que casen las dos imágenes antes de todo habrá que diseñar un método por el que sacar los puntos de las imágenes que se deberán rotar y trasladar. Y, después, utilizando el algoritmo CLONALG implementar una solución para que estos puntos lleguen a coincidir o se queden lo más próximos posible a los de la imagen tomada como modelo.

Por ello primero se debe realizar un pre-procesamiento de la información de la que se dispone para obtener los datos estrictamente necesarios y obviar aquellos que no lo sean. Y, a continuación, se debe desarrollar una implementación del algoritmo que utilice estos datos y de la mejor solución posible, de forma que se logre el scan matching entre un par de imágenes.

Como ya se ha visto los algoritmos evolutivos, en concreto el CLONALG, tiene una serie de parámetros que se pueden modificar para alcanzar la solución óptima. Aquí, en primer lugar, se mostrará el estudio que se ha realizado de cada uno de estos parámetros y se reflejarán, tanto cuantitativamente como cualitativamente, sus variaciones. Una vez estudiado el alcance de cada parámetro del CLONALG se mostrará la solución encontrada que mejor resuelva nuestro problema de optimización.

6.2. DISEÑO DE LA SOLUCIÓN

PRE-PROCESAMIENTO DE LAS IMÁGENES

Para iniciar el scan matching entre un par de imágenes antes de todo se deben obtener estas imágenes y procesarlas para adaptar la información que nos dan a la solución planteada. Computacionalmente el gran número de puntos que forman una imagen es enorme, por ello se deben seleccionar sólo aquellos que resulten verdaderamente útiles y así hacer el programa menos pesado al trabajar con menor cantidad de información. Por eso es necesario realizar este pre-procesamiento en el que se reducirán los puntos de interés a aquellos que presenten diferencias de color más significativas. Se va a utilizar una de las versiones propuestas para este cálculo del CIE y en la que se utiliza la medida Delta E.

De entre todas las versiones propuestas por el CIE para el cálculo de diferencias de color en una imagen, en este trabajo se ha utilizado la versión CIE76 debido a su simplicidad. La ecuación empleada para llevar a cabo este cálculo es:

$$\Delta E^*_{ab} = \sqrt{(L^*_2 - L^*_1)^2 + (a^*_2 - a^*_1)^2 + (b^*_2 - b^*_1)^2} \quad (3)$$

donde $L_1{}^*a_1{}^*b_1{}^*$ y $L_2{}^*a_2{}^*b_2{}^*$ son parámetros de color de dos puntos diferentes y $\Delta E_{ab}{}^*$ da la diferencia cuantitativa de color entre estos dos puntos.

Una vez obtenidos estos puntos de interés ya se puede comenzar a desarrollar el algoritmo que realizará el matching entre imágenes, el CLONALG en este trabajo.

IMPLEMENTACIÓN CLONALG

Como ya se ha visto en el apartado 4.3 el algoritmo CLONALG sigue una serie de pasos en su desarrollo. Para dar solución al problema que se presenta en este proyecto, realizar un matching entre dos imágenes, se va a analizar cada paso detenidamente, explicando los diferentes valores que se han tomado en cada variable y las funciones que se han implementado para llegar a la solución final.

El algoritmo implementado sigue los siguientes procesos:

1. Generación de una población inicial.
2. Cálculo de la función de coste para cada elemento de l población inicial.
3. Selección de elementos con mejores valores de la función de coste.
4. Clonación.
5. Mutación.
6. Nuevo cálculo de la función de coste para los elementos mutados.
7. Selección de los mejores elementos para pertenecer al set de memoria.
8. Generación de nuevos elementos para formar parte de una nueva población.

Cuando se llega al octavo paso se repite todo el proceso tantas veces como iteraciones se hayan definido para llegar a la solución óptima.

Antes de iniciar el algoritmo es necesario tener definidos y conocer una serie de parámetros que son las dos imágenes con las que realizar el matching, A y B, el tamaño de población, NP, y el número de iteraciones, iter_max. Estos son los parámetros de entrada del algoritmo y son imprescindibles para que funcione.

Obviamente, antes de nada, el algoritmo necesita tener las dos imágenes con las que se quiere hacer el matching, *A* y *B*, siendo *A* la imagen que se toma como modelo y *B* la que da los datos necesarios para trabajar y manipular hasta lograr el matching entre ellas.

También hay que definir previamente el tamaño de la población con la que trabajará el algoritmo. Este tamaño se define con la variable NP y es el número de elementos que forman la población inicial. Dar un valor demasiado pequeño a este parámetro puede provocar que haya poca diversidad y por lo tanto que el algoritmo tarde más en converger. Por ello conviene dar un valor algo elevado pero tampoco excederse, ya que el coste computacional puede ser muy grande. El tamaño de la población seleccionado para este trabajo ha sido de 30 individuos ya que da resultados satisfactorios con un coste computacional razonable. En este algoritmo hay que tener especial cuidado con este parámetro, ya que en el proceso de clonación los tamaños de las poblaciones con las que se trabajan se pueden

disparar, incrementando de forma considerable el tiempo de ejecución y perdiendo eficiencia.

Finalmente antes de iniciar el algoritmo se debe determinar el número de iteraciones que realizará. Este será el criterio de parada y al igual que ocurre con el tamaño de la población, es oportuno analizar qué valor debe tomar. Realizar pocas iteraciones puede llevar a que el algoritmo no converja en la solución adecuada, pero, iterar demasiadas veces aumenta mucho el coste computacional. Es por ello que se debe elegir un valor suficiente que garantice que la función se optimiza. Para este caso concreto realizar 50 iteraciones es suficiente para obtener una buena solución, además, experimentalmente se ha comprobado que a partir de las 50 iteraciones el resultado del matching mejora muy paulatinamente, como puede apreciarse en la siguiente tabla:

Iteraciones	Valor función de coste para distintos pares de imágenes			
	50-60	1450-1451	1000-1010	500-501
10	0.310616	0.468719	0.326505	0.235864
20	0.236987	0.427633	0.292780	0.214386
30	0.219173	0.400392	0.278407	0.195912
40	0.213743	0.375491	0.265799	0.188718
50	0.201928	0.360725	0.265799	0.165740
60	0.192548	0.358594	0.262535	0.159877
70	0.186101	0.355126	0.259824	0.156152
80	0.171231	0.349832	0.256156	0.146788
90	0.171231	0.349832	0.250508	0.140405
100	0.171231	0.349832	0.250508	0.139711

TABLA 1.- VALORES FUNCIÓN DE COSTE PARA DIFERENTES PARES DE IMÁGENES.

Estos resultados pueden variar de una ejecución a otra debido a la aleatoriedad con la que trabaja el CLONALG, pero dan una idea de cómo afecta modificar el número de iteraciones. Como se puede observar no merece la pena el coste computacional que supone aumentar las iteraciones con la mejora de los resultados obtenidos.

Una vez determinados los parámetros de inicialización del algoritmo, ya se puede analizar el funcionamiento de éste detenidamente. Para ello se dividirá en los siguientes procesos:

- Población inicial.
- Función de coste o función objetivo.
- Selección.
- Clonación.
- Mutación.
- Conjunto de memoria.

El análisis de cada punto se realiza a continuación para explicar su funcionamiento detenidamente y las decisiones tomadas respecto a algunas funciones y parámetros empleados.

POBLACIÓN INICIAL

Para comenzar la implementación del algoritmo se necesita partir de una población inicial, que es equivalente al conjunto de anticuerpos del que se habla en la teoría de selección clonal. Este conjunto llamado población se crea de forma aleatoria y es de tamaño *NPx(D+1)*, es decir, se crea una población de *NP* individuos con *(D+1)* elementos.

El tamaño de la población viene dado por *NP*, parámetro de entrada del algoritmo y como ya se ha visto es un valor clave para el correcto funcionamiento del algoritmo.

Cada individuo de la población es un vector de *D* elementos. En este caso de estudio el tamaño de este vector es de 6, un valor para coordenada del espacio. Así se trabajará con las tres coordenadas lineales y las otras tres angulares y todas forman el espacio tridimensional que necesitamos para realizar el matching entre imágenes.

Esta población inicial se genera mediante la siguiente función,

$$\mathrm{pop} = \mathrm{inicio\_pop\_slam}(\mathrm{NP}, \mathrm{D}, \mathrm{D\_ang})$$

donde los parámetros de entrada son *NP*, número de individuos de la población, *D*, elementos de cada individuo de la población (6 en este caso para representar las seis coordenadas del espacio tridimensional) y *D_ang*, número de coordenadas angulares y que es igual a tres. La función da como resultado pop, que es una matriz de *NPx(D+1)* (para el caso que nos atañe 30x7).

Esta función fue desarrollada por *Fernando Martín Monar* para el trabajo titulado, *"Analyzing the influence of color in DE-based Scan Matching"*.

En la función básicamente se generan dos vectores, con tres elementos cada uno. El primero representa las variables lineales y estas toman un valor aleatorio dentro del intervalo (-2.5, 2.5). El segundo vector se utiliza para almacenar las tres variables angulares que toman valores aleatorios entre el intervalo (-0.15, 0.15). Además a la población se le añade una séptima columna que se reserva para almacenar el valor de la función de coste que se calcula posteriormente.

FUNCIÓN OBJETIVO O FUNCIÓN DE COSTE

Con la población inicial ya generada ahora llega el momento de calcular el valor de la función objetivo o función de coste. Esta función nos va a dar los valores a optimizar. Como estamos trabajando en un problema de optimización y no de reconocimiento de patrones no tendremos la población de antígenos que aparece en el principio de selección clonal, si no esta función, que nos da los valores con los que el algoritmo trabajará en los sucesivos pasos.

Para ello primero deben corregirse los datos de la población que se han obtenido de forma aleatoria mediante una función llamada *data_correction_by_pose*. Esta función fue desarrollada e implementada por Fernando Martín Monar para el trabajo *"Analyzing the influence of color in DE-based Scan Matching"*.

Al aplicar *data_correction_by_pose* se trasladan y rotan los elementos de la imagen B de acuerdo con los de la población aleatoria obtenida en el paso anterior mediante la función *inicio_pop_slam*. La rotación es dada por la matriz ortonormal con los ángulos de Euler. Como resultado se obtiene una nueva matriz con los datos corregidos.

A continuación, trabajando con los datos corregidos y la imagen usada como modelo se obtiene el valor de la función de coste entre las dos. Para ello se utiliza la función *cost.* Esta función también ha sido desarrollada para un trabajo anterior, *"Analyzing the influence of color in DE-based Scan Matching"*. En este trabajo se crearon diferentes funciones de coste, unas teniendo en cuenta la distancia entre vecinos cercanos y otras añadiendo información del color. En total siete funciones de coste diferentes se desarrollaron utilizando las siguientes técnicas:

1. Distancia entre puntos vecinos calculada punto a punto.
2. Distancia entre puntos vecinos calculada punto a plano.
3. Búsqueda de puntos de acuerdo a las diferencias de color.
4. Distancias ponderadas teniendo en cuenta diferencias de color.
5. Utilizando la matriz de covarianza.
6. Trazando la matriz de covarianza del color.
7. Basándose en NDT (Transformación de distribución normal).

Para este trabajo se ha utilizado la primera versión, que calcula la función de coste para la distancia que hay punto a punto entre correspondientes puntos. Esta distancia viene dada por la siguiente función:

$$e = \sum_{c=1}^{C} d\left(m_{ic}, d_{jc}\right)^2 \quad (4)$$

donde $d\left(m_{ic}, d_{jc}\right)^2$ es la distancia que hay entre dos puntos y C es el número de correspondencias.

Una vez obtenido el valor de la función de coste se almacena en la primera columna de la población para trabajar con él en los siguientes pasos. Este valor es menor cuanto menor es la distancia entre imágenes y por lo tanto hay que lograr minimizarlo. Hay que tener en cuenta que en determinados casos debido a la selección de puntos de las imágenes que se realiza en el preprocesamiento de estas y dependiendo del número de puntos comunes que tengan las dos imágenes este valor puede ser algo mayor y no llegar a minimizarse tanto como en otros casos más favorables.

SELECCIÓN

Esta fase inicial de selección es fundamental en el CLONALG para acotar el número de elementos con los que trabajar. Como este algoritmo ya de por sí tiene un coste computacional elevado se debe hacer una criba inicial para trabajar sólo con los elementos más representativos utilizando un determinado criterio.

El criterio con el que se va a realizar esta selección de los mejores elementos es elegir aquellos que tengan menor función de coste. El número de elementos seleccionados va a ser *n*, que se ha determinado mediante la siguiente ecuación:

$$\mathrm{n} = \mathrm{round}(0.3 \cdot \mathrm{NP}) \qquad (5)$$

donde *NP* es el tamaño de la población, *0.3* es un valor elegido experimentalmente y *round* es el operador de redondeo para que nos dé un número exacto de individuos con los que trabajar.

Este valor también se puede asignar como una constante, pero para hacer más flexible el código se determinará con esta función que le asigna un valor dependiendo del tamaño de la población, es decir, el valor de *n* será proporcional al de la población. Con esto se evita que si la población con la que se quiere trabajar es muy pequeña *n* no sea mayor que *NP* o prácticamente *NP*; y por el contrario, si la población fuera muy elevada y *n* tuviera un valor fijo igual se quedaba muy pequeño y los elementos seleccionados serían insuficientes para ese caso de estudio.

Por lo tanto el proceso a seguir para realizar la selección es: primero, ordenar la población inicial según el valor de la función de coste, es decir, ordenarla de costes menores a mayores; a continuación, calcular el valor de *n* para el caso de trabajo; y, finalmente, seleccionar los *n* mejores elementos de la población ordenada.

En este caso se ha decidido seleccionar el 30% de los elementos de la población. Así, para una población de 30 individuos, como es el caso en el que se trabaja, *n* tendría un valor de 9. Hay que detenerse y reflexionar un poco a la hora de estipular este parámetro, ya que el CLONALG es un algoritmo que va aumentando el tamaño de las poblaciones con las que trabaja, y el coste computacional se puede disparar. Como ya se ha comentado en apartados anteriores una vez realizado este proceso de selección se pasa al de clonación, en el que los *n* mejores elementos se multiplican de acuerdo al valor de la función de coste, según ecuación (6).

Es decir, estos *n* elementos seleccionados en este paso se van a multiplicar en pasos sucesivos aumentando el número de individuos con los que se trabajará. Por lo tanto elegir un valor muy elevado de *n* puede hacer que el algoritmo se vuelva muy pesado computacionalmente.

En cuanto a la influencia que tiene el valor de *n* sobre el número de iteraciones necesarias para que la función converja en un mismo valor, en el trabajo de De Castro and Von Zuben, se observó que es mínima como puede apreciarse en la siguiente gráfica:

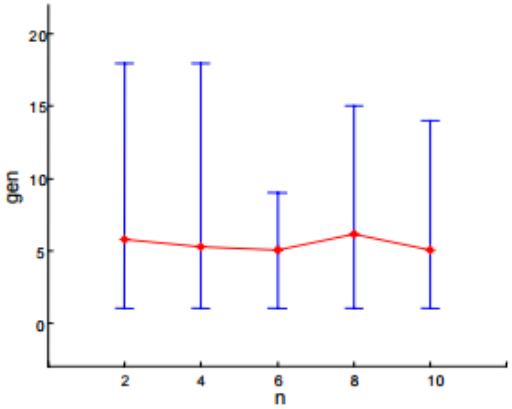

FIGURA 11.-RELACIÓN NÚMERO DE CLONES CON ITERACIONES REALIZADAS EN EL CLONALG.

CLONACIÓN

El proceso de clonación es uno de los principales y más representativos del algoritmo CLONALG. Consiste en crear una nueva población de clones a partir de los *n* candidatos seleccionados en el paso anterior. Estos individuos seleccionados se multiplican proporcionalmente con su afinidad, es decir, los elementos con mejores afinidades (valores más bajos) se clonan más que los elementos con peor afinidad. Así se obtiene una población de clones con *Nc* elementos. Para hallar el número de clones que se deben obtener de cada clon se emplea la siguiente ecuación:

$$N_c = \sum_{i=1}^{n} \text{round}\left(\frac{\beta \cdot NP}{i}\right) \quad (6)$$

donde β es un factor multiplicativo, *NP* es el tamaño de la población, *n* es el número de elementos seleccionados para ser clones y *round* es el operador de redondeo utilizado para obtener un valor entero. De esta forma como *n* es un conjunto ordenado de clones según el valor de la función de coste aumente el número de clones que se genera, *Nc*, diminuye, y así se obtienen más clones para elementos mejores y menos clones para los peores.

Como se puede observar este proceso hace que la población aumente en mayor o menor medida dependiendo del valor que se le dé a beta. Este factor hace que el número de clones generados a partir de un mismo clon aumente o disminuya. Cuanto mayor sea beta mayor número de clones se generarán y cuanto menor, menos clones se obtendrán. Por ello hay que determinar un valor que obtenga la cantidad adecuada de clones y no producir en exceso o en defecto. Este valor se ha elegido como 0.5 experimentalmente., ya que para *NP* individuos que forman la población inicial (30 en este caso) con el número de *n* elementos seleccionados (9 individuos para este caso) se obtiene una población de 44 clones, tamaño adecuado para trabajar sin problemas. Si se aumenta, por ejemplo a la unidad, se obtendría un total de 85 individuos, lo cual hace que el coste computacional sea mucho mayor al trabajar con una población casi el doble de grande y los resultados obtenidos no muestran mejoras considerables. Si por el contrario pasa a ser un valor muy pequeño, como 0.1, el número de clones generados sería escaso y a la hora de llevar a cabo el siguiente paso, que es la mutación, habría menores probabilidades de obtener elementos mejores y por lo tanto de optimizar la función de coste.

MUTACIÓN

La mutación es la fase más compleja en todo el desarrollo del algoritmo. A algunos de los individuos ya seleccionados y clonados se les somete a una serie de modificaciones aleatoriamente para variar la diversidad de la población. Estas modificaciones se realizan conforme al valor de la función de coste de cada individuo. Lo elementos que tengan mejores resultados, es decir, el valor de la función de coste sea menor, deben mutar menos, y los que tengan un valor de la función de coste mayor deberán mutar más.

Este proceso se puede separar en dos partes, una primera en la que se calcula una tasa de mutación para cada individuo y así mutar más a los individuos dependiendo del valor de su función de coste; y otra parte en la que se realiza la mutación de cada uno de los parámetros que conforman cada individuo. Finalmente se vuelve a calcular el valor de la función de coste en los elementos mutados para pasar al siguiente proceso en el que se vuelve a hacer una selección de los mejores elementos de la población.

La tasa de mutación es un parámetro que sirve para seleccionar los elementos que serán clonados y los que no. Esta tasa tiene en cuenta el valor de la función de coste, ya que los elementos con peores costes deben mutar más y los elementos con un coste mejor tienen que mutar menos para no perderlos y empeorar el proceso de optimización. Esta tasa de mutación se calcula mediante la siguiente expresión.

$$\text{Tasa\_mut} = e^{-\rho/\text{coste}} \qquad (7)$$

donde ρ es un factor multiplicativo y coste es el valor de la función de coste para el individuo en cuestión. Como el coste para los mejores elementos es menor que para los peores, el valor de la tasa de mutación debe seguir este criterio, ser menor para elementos más favorables y mayor para los más desfavorables. Por ello al calcular el valor de la tasa de mutación el valor del coste está dividiendo y no multiplicando, como ocurre en otros problemas de optimización.

El valor de ρ se estimó experimentalmente y se observó que no se podía mantener constante en las repetidas iteraciones del algoritmo ya que convergía prematuramente dando resultados poco satisfactorios. Por ello se desarrolló un método por el que el valor de ρ fuera disminuyendo con las sucesivas iteraciones y así lograr que el algoritmo funcionara mejor. La función utilizada para lograr este propósito es la siguiente:

$$\rho_i = \rho_{i+1} - \frac{\rho_i}{iter\_max} \quad (8)$$

donde iter_max es el valor de iteraciones máximas fijadas que realiza el algoritmo (50 para este trabajo) y ρ va variando progresivamente según aumenta el valor de i, que es la iteración actual. Como se puede observar el valor de ρ va disminuyendo lo que hace que el valor de la tasa de mutación aumente ligeramente según aumenta el número de iteración. Este aumento progresivo es importante debido a que el valor de la tasa de mutación se utiliza en el siguiente paso que es la propia mutación.

Para mutar, como ya se verá un poco más adelante, se va sumando un pequeño valor a cada uno de los elementos que conforman un individuo. Este valor lo da la tasa de mutación. Según se progresa en las iteraciones, como es de esperar, el algoritmo va consiguiendo resultados menores, y por lo tanto el valor de la función de coste va disminuyendo. Esto hace que también vaya disminuyendo el valor de la tasa de mutación y, por tanto, la mutación también disminuye. Este comportamiento no es apropiado y produce una convergencia prematura del algoritmo ya que a partir de unas iteraciones la mutación es tan pequeña que raramente se obtiene algún elemento mutado mejor que el anterior.

Una vez obtenido el valor de la variable ρ se calcula la tasa de mutación y se comienza con el propio proceso de mutación. Antes de que un elemento mute debe pasar una condición, para así lograr que los peores elementos muten más y los mejores menos. Para ello se comprueba el valor de la tasa de mutación con otro valor llamado r y que es uno de los valores elegidos aleatoriamente entre el rango de la tasa de mutación que aparece para la población actual. Este valor se calcula para cada clon, así cada uno cumple la condición independientemente. Si no se operara de este modo, como en la población de clones cada elemento está repetido un cierto número de veces, según el clon del que se trate, el requisito para mutar sería el mismo para individuos procedentes de un mismo clon, y se mutarían todos o ninguno de los elementos generados de un clon. A continuación se muestra en una tabla como varía el número de clones mutados para cada iteración:

Iteración	Número de clones mutados
1	24
2	10
3	16
4	14
5	18
6	15
7	11
8	14
9	13
10	16

TABLA 2.- NÚMERO DE CLONES GENERADOS EN DIFERENTES ITERACIONES.

El número de individuos que mutan es de vital importancia para el correcto funcionamiento del algoritmo. Si este número es muy pequeño se tardará mucho o incluso no se llegará a la solución adecuada. Por el contrario, si este valor es muy elevado se podría llegar a perder buenos candidatos, retrasando también la convergencia. Lo ideal es que el número de clones sea el suficiente para introducir la variedad justa en la población de clones. Tras un proceso experimental se observó que el número de clones que se mutan con el método anteriormente descrito es el correcto y suficiente para que el algoritmo funcione bien.

A continuación se pasa al propio proceso de mutación si se ha determinado que el elemento de la población de clones con el que se está trabajando deba mutar. Este proceso es clave en el CLONALG. Se han desarrollado varias funciones de mutación a lo largo de los años, atendiendo a los diferentes tipos de problemas para los que se aplica este algoritmo. En este trabajo se han analizado cuatro de ellos, estudiando cuál responde mejor al problema estudiado.

Estos cuatro operadores de mutación parten todos de uno inicial y van añadiendo modificaciones que los hacen más adecuados para unos casos u otros. El operador de mutación inicial es:

$$C_{mut}(i,j) = C(i,j) \pm tasa\_mut * rang \qquad (9)$$

donde Cmut es el elemento mutado, C es el elemento en su estado inicial, sin mutar, tasa_mut es la tasa de mutación calculada en el paso anterior y según la expresión (x) y rang es un valor elegido aleatoriamente de los que conforman el rango de cada componente de cada individuo.

La siguiente expresión empleada para el cálculo de la tasa de mutación es una pequeña variación de la anterior:

$$C_{mut}(i,j) = C(i,j) \pm \frac{tasa\_mut * rang}{iter\_actual} \quad (10)$$

En esta expresión al elemento original se le añade una pequeña variación que viene dada por la tasa de mutación, rang (valor elegido aleatoriamente de entre todos los presentes en el rango de un elemento) e iter_actual, que da el valor de la iteración actual en la que se está trabajando. Como con cada iteración el valor de la variación de la mutación se divide entre ésta en esta variación del operador de mutación se consigue que según el algoritmo progrese la mutación sea menor ya que se va acercando a la solución óptima. Las siguientes versiones de los operadores de mutación son una modificación de esta última, por lo que siguen este comportamiento.

Otro operador de mutación ampliamente utilizado incluye entre sus factores el número de clones, Nc. Su expresión es la siguiente:

$$C_{mut}(i,j) = C(i,j) \pm \frac{tasa\_mut * rang * Nc}{iter\_actual} \quad (11)$$

En este caso al multiplicar por un valor elevado como Nc se consigue que la mutación sea más agresiva lo cual para algunos casos de aplicación puede ser positivo e incluso necesario.

Finalmente el cuarto operador de mutación es una variación del anterior. El número de clones en lugar de aparecer multiplicando está dividiendo, lo que hace que el valor con el que se modifica al individuo disminuya y la mutación sea menos agresiva. Su ecuación es la siguiente:

$$C_{mut}(i,j) = C(i,j) \pm \frac{tasa\_mut * rang}{iter\_actual * Nc} \quad (12)$$

El operador que mejor se ajusta al problema planteado en este trabajo es el primero de ellos, ecuación (X). Utilizando los demás no se llega a una solución adecuada debido a que o se va disminuyendo la mutación según se aumentan las iteraciones, o se realizan mutaciones muy agresivas que se separan mucho de la solución ideal. Para cada uno de las componentes de cada individuo, las tres coordenadas lineales y las tres angulares, se calcula un valor mediante el operador de mutación. Por lo tanto rang será en cada caso un valor perteneciente al rango de cada parámetro elegido aleatoriamente.

Una vez modificados aquellos individuos seleccionados es necesario volver a calcular su valor de la función de coste para poder continuar con el último paso del CLONALG.

CONJUNTO DE MEMORIA

El último paso que se realiza en el algoritmo CLONALG es el proceso de creación del conjunto de memoria. Es este proceso se vuelve a hacer una selección de los mejores individuos que ya han sido clonados y mutados anteriormente para que pasen a formar parte de un conjunto de memoria.

Este grupo de elementos, como ya se explicó en la teoría de selección clonal, almacena los mejores individuos para tenerlos en iteraciones futuras y cada vez partir de mejores individuos. Una vez seleccionados estos elementos se vuelve a crear otra población añadiendo d elementos.

El número de elementos que se seleccionan para este caso viene representado por m. Experimentalmente se le ha determinado un valor fijo bastante reducido, se seleccionarán tan sólo los tres mejores elementos. Esto es debido a que se observó que en las primeras posiciones de la población de clones mutados había poca diversidad y al final en este paso se seleccionaban muchos elementos iguales.

Para obtener los tres mejores de toda la población de clones mutados se ordenan según los valores de su función de coste, de forma que los primeros sean los mejores, es decir, tengan un valor menor de coste, y los últimos los peores. Estos últimos se descartarán y en su lugar se insertará una nueva población de Nd individuos. Con la idea de mantener el número de elementos de una población constante iteración tras iteración el valor de Nd será calculado de forma que junto con los Nm elementos seleccionados sean igual a NP, es decir:

$$NP = N_m + N_d \qquad (13)$$

Es importante mantener el tamaño de la población inicial debido a que como se ha visto el CLONALG es un algoritmo que multiplica el tamaño de las poblaciones con las que trabaja, y si no se acota al pasarlas de iteración en iteración los tamaños pueden ir creciendo rápidamente haciendo el algoritmo poco eficiente debido al alto coste computacional.

La población de Nd elementos es generada y corregida como la población inicial. De este modo en cada iteración se obtiene una nueva población de NP individuos, en la que ya aparecen los Nm mejores elementos de la iteración anterior y se vuelve a iniciar todo el proceso.

7. RESULTADOS EXPERIMENTALES

7.1. RESULTADOS EXPERIMENTALES

Para observar el comportamiento del CLONALG a la hora de realizar el scan matching entre cada par de imágenes elegido se va a mostrar por un lado unos resultados numéricos que cuantifican y verifican que el algoritmo está funcionando correctamente. Por otro lado para cada par de imágenes se va a incluir una gráfica en la que se observa la diferencia final que hay entre las imágenes. Cada una se representará con una nube de puntos (los puntos más significativos con los que se ha trabajado) de diferente color, verde y rojo, para diferenciarlas adecuadamente.

Se han seleccionado un total de cuatro pares de imágenes diferentes para ver cómo actúa el algoritmo. Estos pares son los 50-60, 1450-1451, 1000-1010 y 500-501. Son los mismos pares que se seleccionaron en el trabajo [1] en el que se utilizaba el algoritmo DE para poder comparar las diferentes soluciones que dan estos algortimos.

Imágenes 50-60

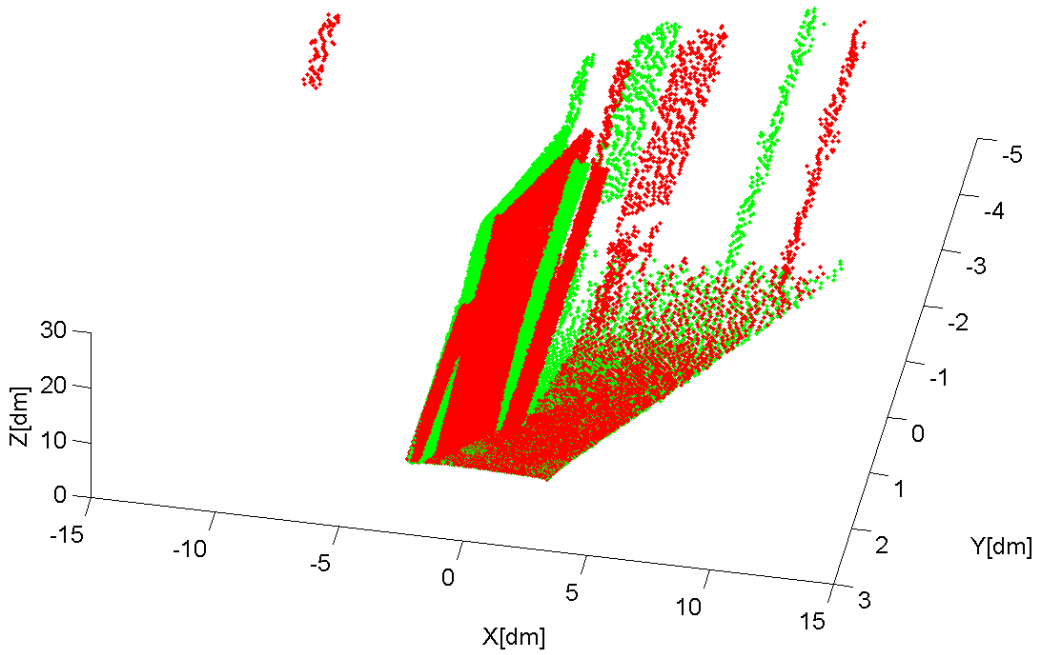

FIGURA 12.- SCAN MATCHING ENTRE IMÁGENES 50-60.

Resultado obtenido función de coste en cada iteración	
Iteración	**Coste**
5	0.404997
10	0.312016
15	0.264899
20	0.251567
25	0.238714
30	0.197486
35	0.192666
40	0.192666
45	0.192666
50	0.192666

TABLA 3.- VALORES FUNCIÓN DE COSTE PARES DE IMÁGENES 50-60.

Imágenes 1450-1451

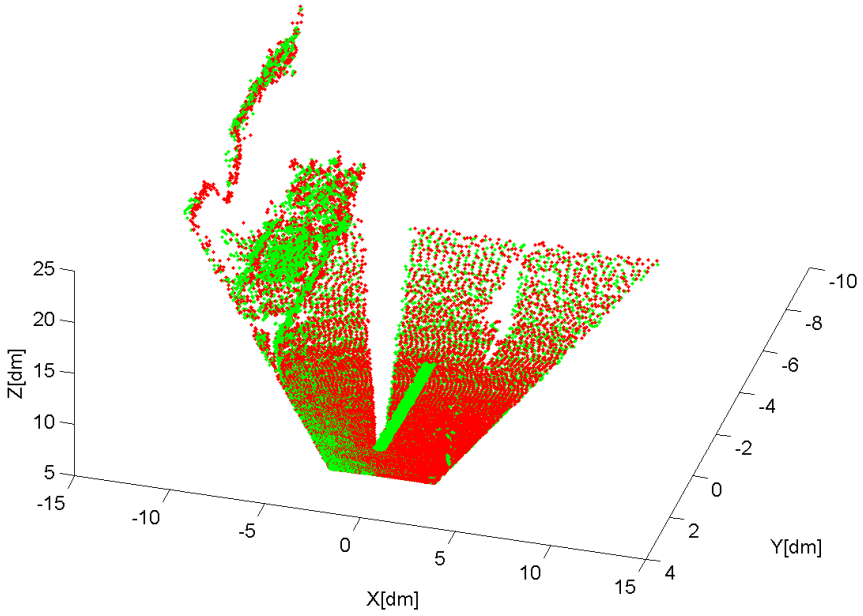

FIGURA 13.- SCAN MATCHING ENTRE IMÁGENES 1450-1451.

Resultado obtenido función de coste en cada iteración	
Iteración	**Coste**
5	0.637591
10	0.494248
15	0.395793
20	0.373163
25	0.359565
30	0.359565
35	0.359565
40	0.335733
45	0.335733
50	0.335733

TABLA 4.- VALORES FUNCIÓN DE COSTE PARES DE IMÁGENES 1450-1451.

Imágenes 1000-1010

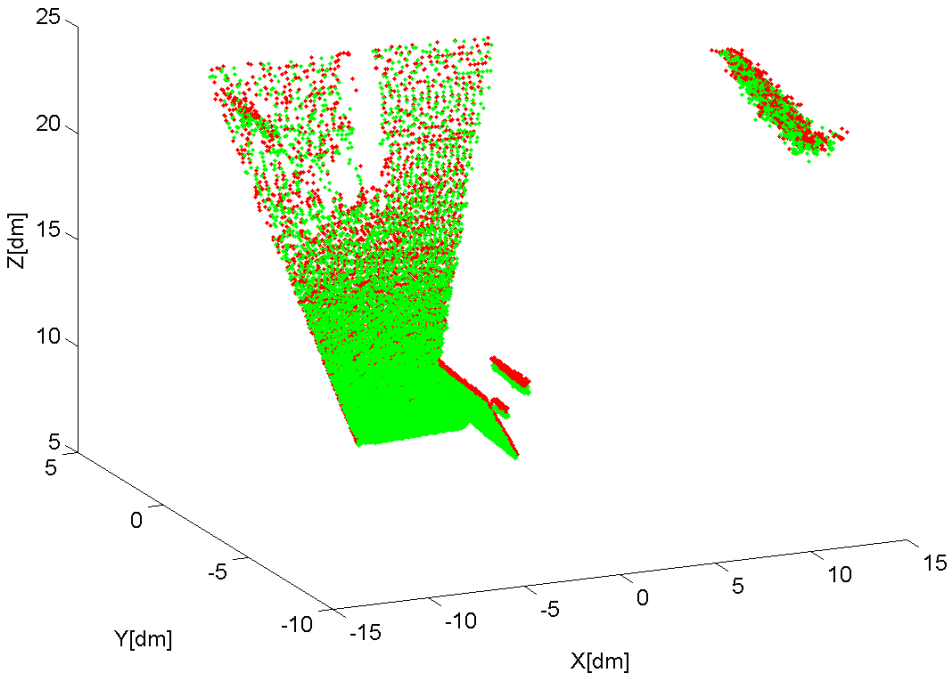

FIGURA 14.- SCAN MATCHING ENTRE IMÁGENES 1000-1010.

Resultado obtenido función de coste en cada iteración	
Iteración	**Coste**
5	0.493782
10	0.397877
15	0.265612
20	0.260706
25	0.258118
30	0.255254
35	0.255254
40	0.255254
45	0.255254
50	0.254800

TABLA 5.- VALORES FUNCIÓN DE COSTE PARES DE IMÁGENES 1000-1010.

Imágenes 500-501

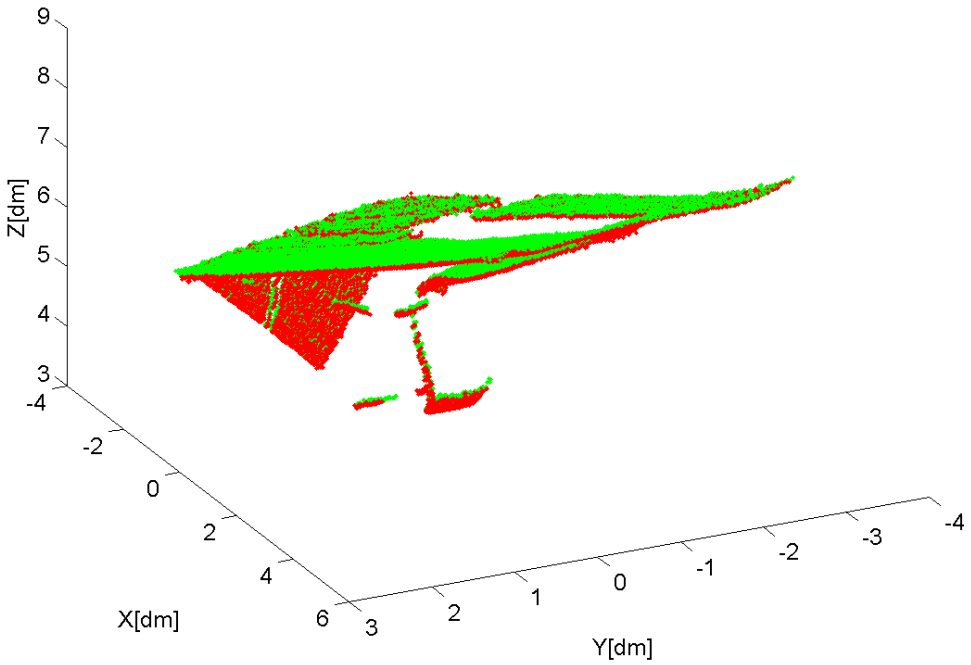

FIGURA 15.- SCAN MATCHING ENTRE IMÁGENES 500-501.

Resultado obtenido función de coste en cada iteración	
Iteración	**Coste**
5	0.378865
10	0.247186
15	0.205835
20	0.199016
25	0.198417
30	0.198417
35	0.198417
40	0.193950
45	0.125369
50	0.121806

TABLA 6.- VALORES FUNCIÓN DE COSTE PARES DE IMÁGENES 500-501.

Como puede apreciarse en las tablas de los resultados obtenidos el valor de la función de coste va mejorando iteración tras iteración, lo que indica que el algoritmo funciona correctamente. Se puede apreciar como cuando el resultado que se obtiene se acerca al óptimo se producen menos variaciones y cuesta cada vez más seguir mejorando el resultado, lo cual es completamente lógico.

Por otro lado en las imágenes mostradas se puede ver cómo quedan los puntos de interés una vez optimizados. Hay pares de imágenes en las que el resultado es mejor que en otras. Esto es debido a la selección de puntos que se hace y a los puntos en común que tengan.

Como es de esperar, imágenes más distantes tendrán un resultado peor debido a que tienen menos puntos en común, como ocurre en el par 60-70, por ejemplo. En cambio en imágenes próximas realizar el matching es más sencillo debido a la enorme cantidad de puntos que tienen en común, como se puede ver en el par 500-501. Este comportamiento también se puede apreciar claramente en los valores que nos dan las tablas que muestran la función de coste. Imágenes más distantes parten de un valor de coste mayor el cual será más difícil de optimizar.

7.2. ANÁLISIS DEL VALLE DE CONVERGENCIA

El análisis del valle de convergencia es una de las pruebas que se realizan a la hora de implementar un método de scan matching y que sirve para comprobar la eficiencia de éste.

Este procedimiento consiste en incorporar un offset a la posición y rotación de las imágenes y observar los resultados obtenidos para ver si se produce la alineación de las imágenes con estos offsets o no.

En este trabajo se han introducido offsets de traslación y rotación en el plano horizontal, ya que se asume que un robot tan sólo se desplaza por éste y no por el vertical. Respecto a la traslación se ha introducido un offset que va en el intervalo [-2,2] metros. En la rotación el intervalo empleado es de [-80, 80] grados. Para cada offset de traslación introducido se ha recorrido todo el intervalo de offsets de rotación. A continuación se muestra en dos tablas los valores de estos offsets.

Offset de traslación
(-20, 0, 20)
(0, 0, 20)
(20, 0, 20)
(-20, 0, 0)
(0, 0, 0)
(20, 0, 0)
(-20, 0, -20)
(0, 0, -20)
(20, 0, -20)

TABLA 7.- OFFSET TRASLACIÓN.

Offset de rotación
(0, -80, 0)
(0, -60, 0)
(0, -40, 0)
(0, -20, 0)
(0, 0, 0)
(0, 20, 0)
(0, 40, 0)
(0, 60, 0)
(0, 80, 0)

TABLA 8.- OFFSET ROTACIÓN.

Una vez introducidos los respectivos offsets se obtienen los resultados mostrados en las siguientes gráficas. Se ha realizado un análisis del valle de convergencia para cada par de imágenes utilizadas en la muestra de los resultados experimentales. En las siguientes gráficas se muestran los resultados satisfactorios de verde y los erróneos de rojo.

Imágenes 50-60

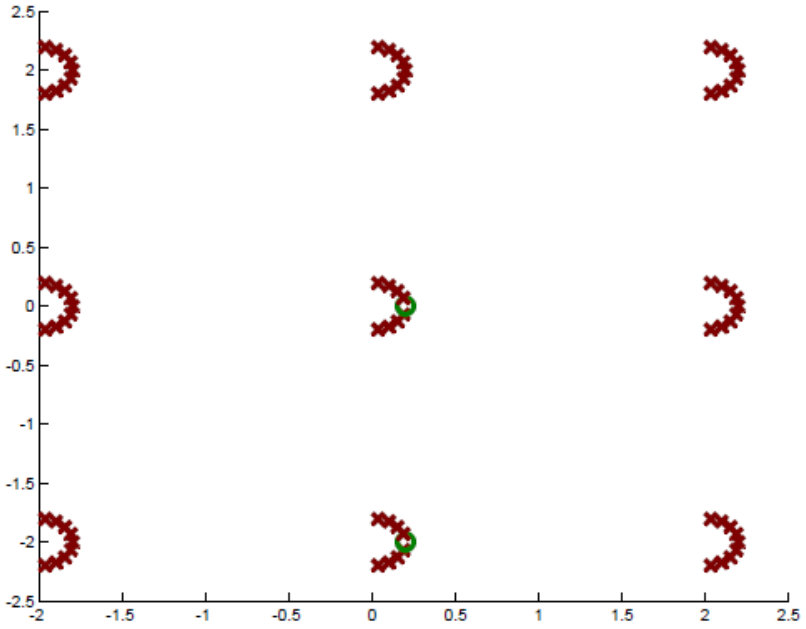

FIGURA 16.- SCAN MATCHING IMÁGENES 50-60 CON OFFSETS DE ROTACIÓN Y TRASLACIÓN.

Imágenes 1450-1451

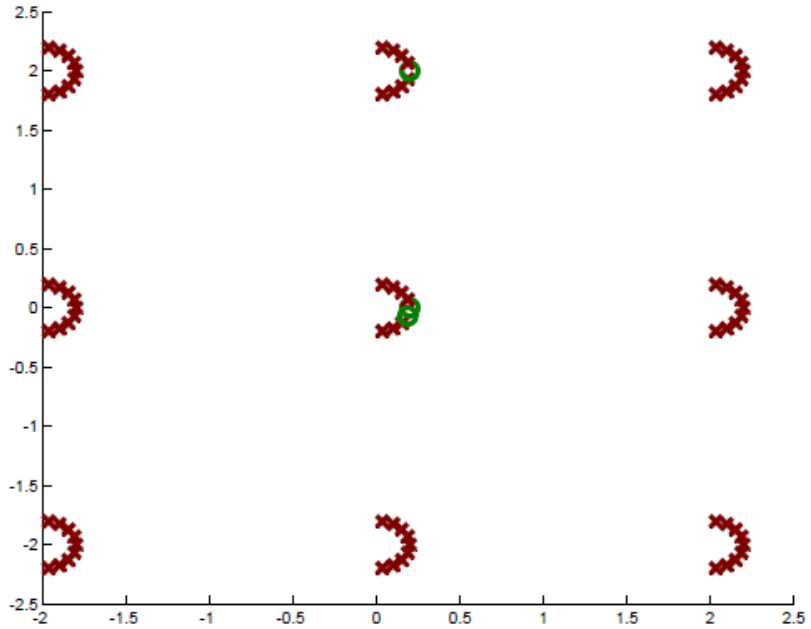

FIGURA 17.- SCAN MATCHING IMÁGENES 1450-1451 CON OFFSETS DE ROTACIÓN Y TRASLACIÓN.

Imágenes 1000-1010

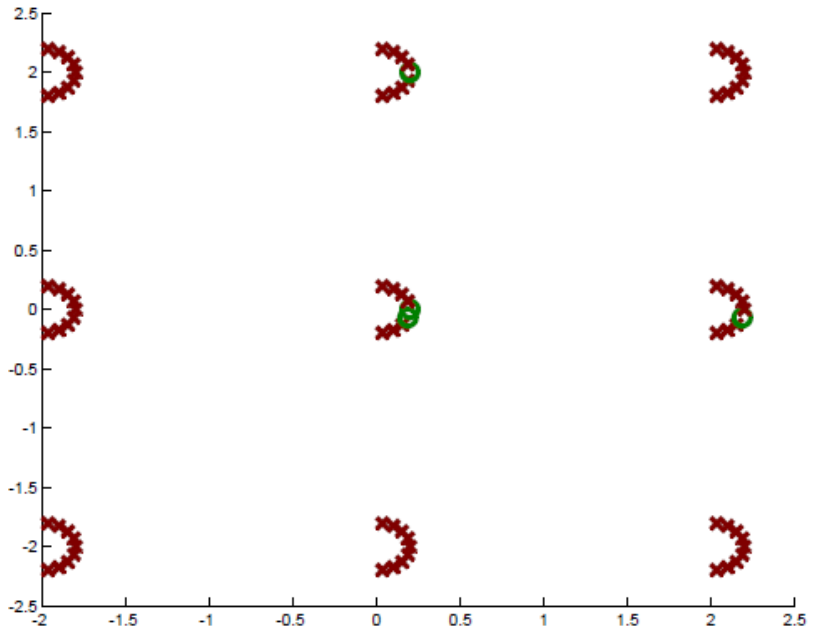

FIGURA 18.- SCAN MATCHING IMÁGENES 1000-1010 CON OFFSETS DE ROTACIÓN Y TRASLACIÓN.

Imágenes 500-501

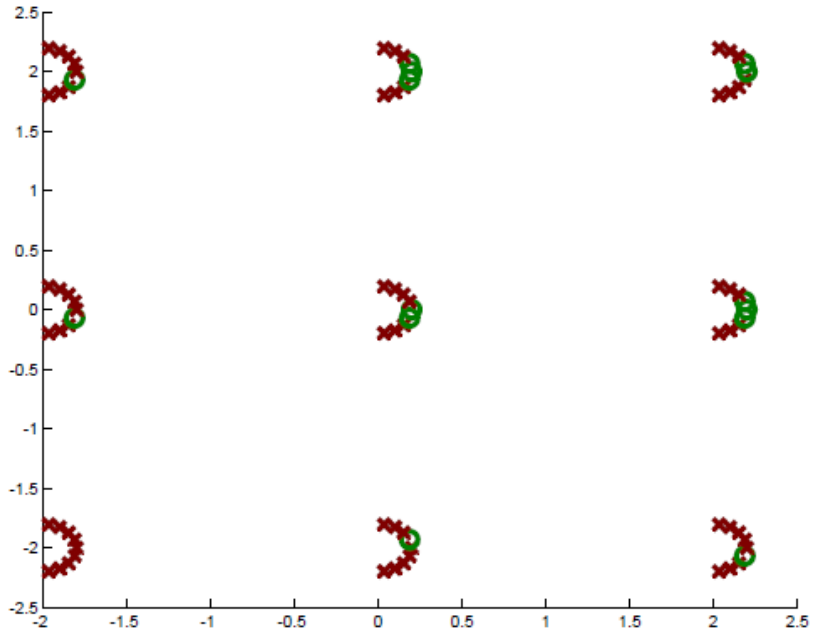

FIGURA 19.- SCAN MATCHING IMÁGENES 500-501 CON OFFSETS DE ROTACIÓN Y TRASLACIÓN.

8. CONCLUSIONES

8.1. CONCLUSIONES

En este trabajo se ha obtenido un nuevo método para solventar el problema del scan matching utilizando para ello el algoritmo CLONALG. Debido a que se trata del mismo problema resuelto en el trabajo de Fernando Martín Monar [1] se han mantenido las mismas condiciones a la hora tanto del desarrollo como de la realización de pruebas experimentales. Así es posible comparar los resultados obtenidos al aplicar diferentes algoritmos. Estos algoritmos son en DE y el CLONALG.

Dichos algoritmos aunque su funcionamiento a grandes rasgos no difiere del propio de los algoritmos evolutivos, presentan diferencias en su desarrollo. Estas diferencias sobre todo son apreciables en los operadores encargados de modificar la población. En el DE se utiliza un operador de cruce, el crossover. Con él se producen nuevos candidatos pero a partir de los iniciales. Y aunque los dos algoritmos realizan una mutación la forma en que se genera es diferente, utilizando para ello operadores de mutación muy distintos.

Otra gran diferencia entre estos dos algoritmos es que en el DE el mejor candidato siempre forma parte de la población con la que se trabaja y no se emplea para generar nuevos candidatos a partir de él. Esta forma de actuación es muy diferente del CLONALG, en el que los mejores candidatos siempre se utilizan para generar a partir de ellos otros candidatos, y, además, tras cada iteración se introducen nuevos individuos generados aleatoriamente que sustituyen a los peores elementos.

Pero estos dos algoritmos también guardan ciertas similitudes en su funcionamiento. Ambos utilizan operadores de selección para discernir entre candidatos propensos a dar una buena solución y los que no.

Respecto a los resultados obtenidos se ha observado como el DE da mejores respuestas que el CLONALG. Esto puede ser debido a dos motivos fundamentalmente. Uno es que el propio funcionamiento de cada algoritmo se adapte mejor al problema presentado y a la forma en la que está presentado. Es

decir, hay que tener en cuenta los datos que se han seleccionado y la forma en la que se utilizan estos datos y analizar y estudiar qué tipo de algoritmo se adapta mejor en cada caso. El otro motivo por el que es posible que el CLONALG haya dado peores resultados es debido al propio desarrollo del algoritmo. Ese desarrollo puede no responder de forma tan eficaz debido al desconocimiento previo del alumno en esta materia. Aunque concluye el trabajo con la capacidad y el conocimiento suficiente como para seguir investigando y desarrollar mejores soluciones.

Esta diferencia en los resultados de los dos algoritmos se puede ver claramente si se compara el análisis del valle de convergencia resultante de este trabajo, y representado en las figuras 16, 17, 18 y 19, con el obtenido para el Differential Evolution en el trabajo realizado por Fernando Martín Monar [1] y representado en la figura 20, que se muestra a continuación,

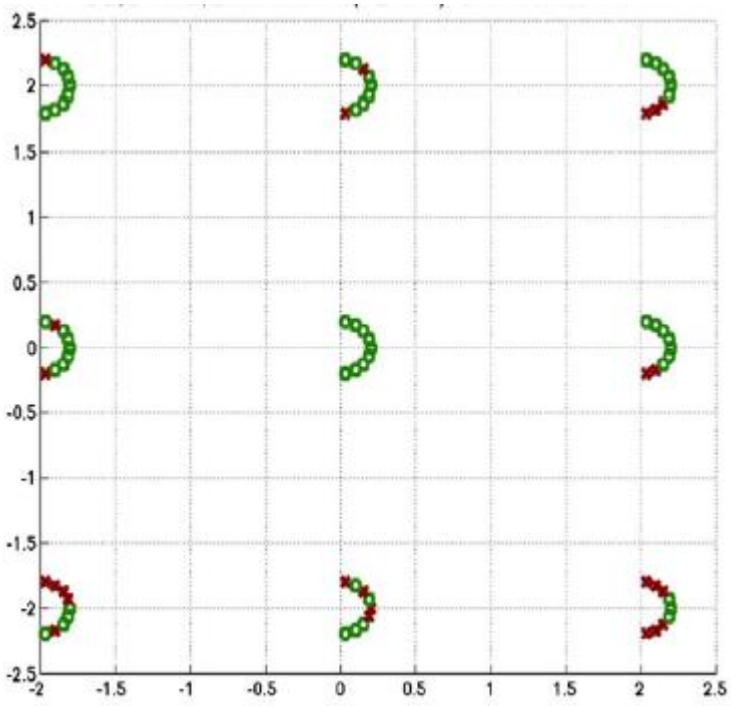

FIGURA 20. VALLE DE CONVERGENCIA PARA RESULTADOS SCAN MATCHING UTILIZANDO DE EN IMÁGENES 1000-1010.

Después de este estudio se ha podido comprobar las dificultades que presentan estos tipos de algoritmos, como:

- Definición del tipo de problema y uso de una representación adecuada.
- Selección de la función de coste a optimizar y que represente de forma precisa el problema descrito.
- Selección de los parámetros que utiliza el propio algoritmo dependiendo del tipo de problema que se quiera resolver.

- Coste computacional, tiempo requerido y dificultad que presentan en comparación con los métodos analíticos.

En la opinión del alumno se han cumplido todos los objetivos que tenía el presente trabajo, tanto personales como técnicos.

8.2. FUTURAS LÍNEAS DE INVESTIGACIÓN

En un futuro, partiendo del problema inicial y la solución obtenida se podría seguir investigando sobre esta materia para obtener mejores soluciones o adaptarlas a otros tipos de problemas.

Debido a que los resultados obtenidos en este trabajo no han sido tan satisfactorios como se esperaban se podría seguir investigando en esta línea para mejorarlos, tanto cualitativamente como cuantitativamente. Como ya se ha visto en el CLONALG se pueden utilizar una gran variedad de operadores de variación, por ejemplo los empleados a la hora de realizar la mutación. Esta libertad hace que desarrollar una solución sea una tarea compleja y, por lo tanto, si se quieren llegar a excelentes resultados se debe hacer un minucioso estudio. También sería interesante estudiar detenidamente el coste computacional del algoritmo y desarrollar soluciones que lo minimicen para hacerlo más flexible a la hora de ponerlo en práctica.

Otra posible línea de investigación sería la sustitución del algoritmo empleado por otros basados también en los Sistemas Inmunológicos Artificiales, como el algoritmo de selección negativa [20] o las redes inmunológicas artificiales [21]. Para la aplicación de dichos algoritmos habría que realizar un estudio propio de los mismos ya que su funcionamiento difiere del CLONALG aunque la base teórica sea la misma.

A. BIBLIOGRAFÍA

[1] Fernando Martín, Jaime Valls Miró, Luis Moreno: "RGB-D DE-based Scan Matching: Exploiting Colour Properties in Registration". *Journal of Intelligent and Robotic Systems with a special section on Unmanned Systems*. Springer, 2014.

[2] Leandro N. de Castro, Fernando J. Von Zuben: "Learning and Optimization Using the Clonal Selection Principle".

[3] https://www.microsoft.com/en-us/kinectforwindows/develop/default.aspx

[4] http://es.mathworks.com/products/matlab/

[5] J. Leonard, H. F. Durrant-Whyte: "Mobile Robot Localization by Tracking Geometric Beacons", *IEEE Transactions on Robotic and Automation*, Vol 7, No 3, pp. 376-382, 1996.

[6] Smith, Randall; Self, Matthew; Cheeseman, Peter: "Estimating uncertain spatial relationships in robotics". *IEEE International Conference on In Robotics and Automation*. Proceedings, 1987.

[7] Optimización (matemática). En Wikipedia, la enciclopedia libre. https://en.wikipedia.org/wiki/Mathematical_optimization

[8] R. Storn; K. Price: "Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces". *Technical Report* TR-95-012, 1995.

[9] K. Price: "An introduction to Differential Evolution". In *New Ideas in Optimization*, McGraw-Hill, London, 1999.

[10] Besl, P.J., McKay, N.D.: "A method for registration of 3d shapes". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992.

[11] Hähnel, D., Burgard, W., Thrun, S.: "Learning compact 3D models of indoor and outdoor environments with a mobile robot". *Robot. Auton. Syst*, 2003.

[12] Triebel, R., Pfaff, P., Burgard, W.: "Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing" In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

[13] Biber, P., Straßer, W.: “The Normal Distributions Transform: A New Approach to Laser Scan Matching” In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[14] Hunter, R.S.: “Photoelectric Color-Difference Meter” In *Proceedings of the Winter Meeting of the Optical Society of America,* 1948.

[15] Hunter, R.S.: “Accuracy, Precision, and Stability of New Photo-electric Color-Difference Meter” In *Proceedings of the Thirty-Third Annual Meeting of the Optical Society of America*, 1948.

[16] J. H. Holland: *Adaptation in Natural and Artificial Systems*. 5 ed. Cambridge, MA: MIT Press, 1998.

[17] T.Bäck. D. B. Fogel, and Z. Michalewicz, Evolutionary Computation 1: *Basic Algorithms and Operators*, Bristol, U.K.: Inst. Of Physiscs, 2000.

[18] F.M. Burnet: “Clonal selection and after,” in *Theoretical Immunology*, G.I. Bell, A. S. Perelson, and, G.H. Pimbley Jr. Eds. New York: Marcel Dekker, 1978.

[19] *The Clonal Selection Theory of Acquired Inmunity*. Cambridge, U.K.: Cambridge Univ. Press, 1959.

[20] S. Forrest, A. Perelson, L. Allen, and R. Cherukuri: “Self-nonself discrimination in a computer”, In *Proc. of the IEEE Symposium on Security and Privacy*, IEEE Computer Society, 1994.

[21] TIMMIS, Jonathan: “Artificial Immune Systems: A novel data analysis technique inspired by immune network theory”. PhD thesis, University of Wales, Aberystwyth. Ceredigion, Wales, 2001.

B. APÉNDICES

B.1. PLANIFICACIÓN

El presente trabajo se ha organizado como se puede observar en el diagrama de Gantt adjunto a continuación. Para ello se ha dividido en una serie de fases:

1. Planteamiento.
    - Familiarización.
    - Búsqueda de la información.
    - Diseño.
2. Desarrollo.
    - Desarrollo algoritmo.
    - Implantación algoritmo.
3. Pruebas.
    - Pruebas.
    - Depuración.
4. Documentación.
    - Redacción.
    - Corrección.

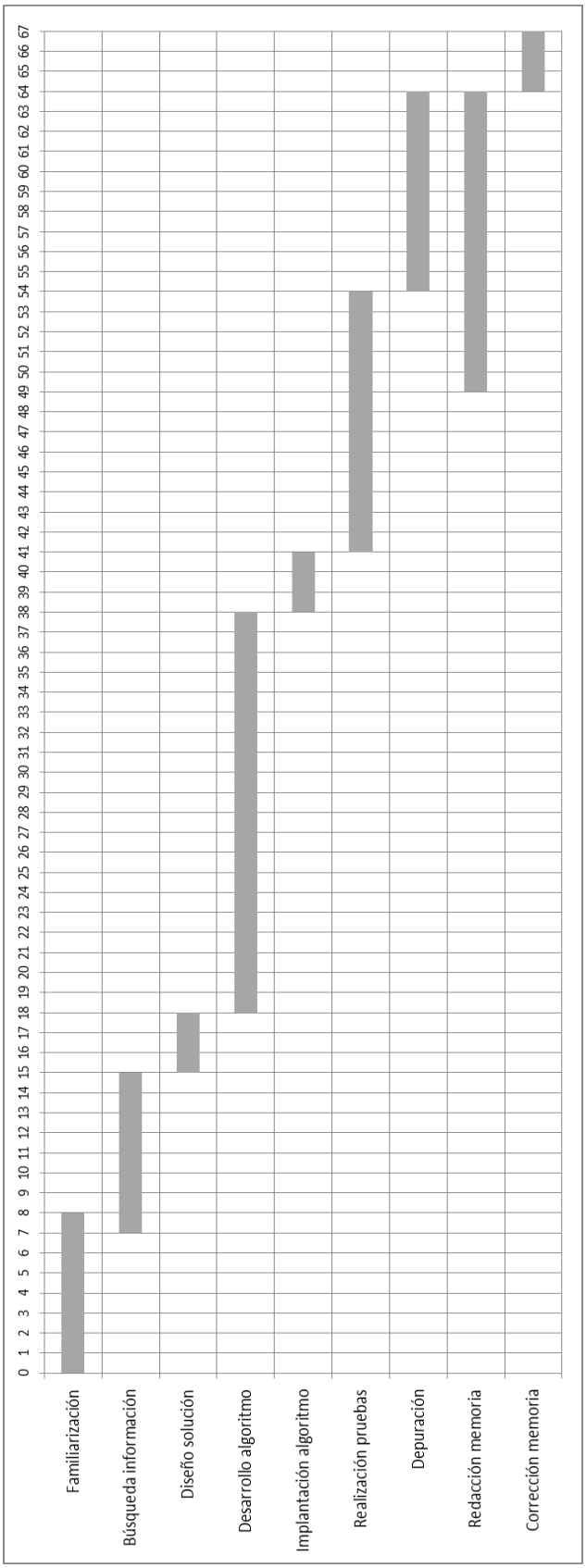

FIGURA 21.- DIAGRAMA DE GANTT.

B.1. PRESUPUESTO

Debido a que este proyecto es de tipo técnico-experimental en el presupuesto se va a reflejar tanto el coste de los equipos empleados en el desarrollo del mismo como el coste aproximado del personal que lo ha llevado a cabo.

COSTE DE MATERIAL

En la siguiente tabla se puede ver el desglose del coste del material requerido:

COSTE MATERIAL			
Familia	**Concepto**	**Coste total**	**Coste proyecto**
Software	Software	500€	500€
Hardware	Ordenador	550€	45€
Otros	Internet	20€/mes	60€
	Material oficina	50€	50€
SUBTOTAL			**655€**

TABLA 9.- COSTE MATERIAL.

COSTES DE PERSONAL

Para calcular el coste del personal se va a estipular en primer lugar el número de horas empleadas tanto por el tutor del proyecto como por el alumno. Este número de horas se reflejarán en la siguiente tabla:

HORAS PERSONAL			
Fase	**Tarea**	**Tutor**	**Alumno**
Planteamiento	Familiarización	0	30
	Búsqueda información	10	30
	Diseño	0	10
Desarrollo	Desarrollo algoritmo	10	80
	Implantación algoritmo	10	10
Pruebas	Pruebas y análisis	0	50
	Depuración	0	40
Documentación	Redacción	0	60
	Corrección	10	10
TOTAL		**40**	**320**

TABLA 10.- HORAS PERSONAL.

Una vez conocido el tiempo empleado por el personal se puede estimar el coste que ha supuesto,

COSTE PERSONAL			
Empleado	**Precio unitario (€/h)**	**Nº horas**	**Importe€**
Tutor	30	40	1200€
Alumno	8	320	2560€
TOTAL			**3760€**

TABLA 11.- COSTE PERSONAL.

COSTE TOTAL

En la siguiente tabla se refleja el coste total del proyecto,

PRESUPUESTO TOTAL	
Concepto	Importe €
Personal	3760.00
Material	655.00
TOTAL	**4415.00**
21% IVA	927.15
TOTAL	**5378.15€**

TABLA 12.- PRESUPUESTO TOTAL.

El presupuesto final ascendería a CINCO MIL TRESCIENTOS SETENTA Y OCHO EUROS CON QUINCE CÉNTIMOS.