# Detección de parábolas en imágenes de arterias mediante algoritmos de estimación de distribución y algoritmos genéticos

Erick Salvador Alvarez Valencia, CIMAT A.C.

***Resumen*—En este proyecto se trató un caso particular de procesamiento de imágenes médicas, las que consisten en arterias de retina. El propósito fue el tratar de detectar una parábola que se ajuste adecuadamente a la imagen. Para ello se recurrió al área de metaheurísticas, ya que en los últimos años este tipo de algoritmos han dado muy buenos resultados en la parte de optimización estocástica.**

**Los algoritmos basados en técnicas evolutivas y de estimación de distribución imitan el proceso de evolución biológica, esto para encontrar mejores soluciones a las ya obtenidas, para lograr esto recurren a los procesos de cruza y mutación de individuos. En el presente trabajo se recurrió a dos algoritmos para encontrar parábolas que se ajusten de manera correcta al modelo, el algoritmo UMDA y un algoritmo genético. Ambos algoritmos usaron el mismo planteamiento de solución pero trabajaron de manera distinta para encontrarla. En el presente informe se mostrará la metodología usada y los resultados que se obtuvieron con cada algoritmo usado, así como las imágenes de salida que estos brindaron.**

***Index Terms*—Detección parábola, arterias, EDA, Algoritmo genético, Visión computacional.**

## I. Introducción

EL procesamiento de imágenes es un área muy activa en computación y bastante aplicada en diversas áreas de la ciencia, tal es el caso de la medicina, en donde diariamente se generan muchas imágenes médicas de diversa índole, ej.: Imágenes de radiología, endoscopía, termografía, etc.

En el presente reporte se describirá el uso de dos algoritmos estocásticos con el fin de la detección de parábolas en imágenes de arterias de retina. Dichos algoritmos son el UMDA y el algoritmo genético, ambos fueron probados y comparados con los mismos parámetros y la misma función de evaluación para ver cuál de ellos generaba mejores resultados.

Noviembre, 2017

## II. Descripción del problema

El procesamiento de imágenes es una rama de la computación que lleva varios años en plena investigación, cada vez se generan nuevos y mejores métodos para realizar diversas tareas en esta área. Una subárea interesante del procesamiento de imágenes es el reconocimiento de patrones, en el caso del presente trabajo el problema presentado es el reconocimiento de buenos modelos de parábola que se adapten a imágenes médicas de arterias de retina.

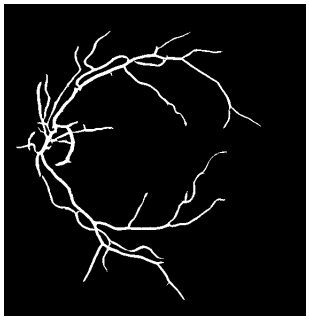
Es del presente problema el interés de dicho reconocimiento ya que de esta forma se podrían detectar conjuntos de retinas en estado enfermo o sano, de la misma manera podemos aplicar este método a otros conjuntos de imágenes como por ejemplo imágenes de pies planos, ya que estos no cuentan con el arco adecuado.

En la actualidad existen varios algoritmos para detectar formas en una imagen, uno de ellos es la tan conocida transformada de Hough, la cual se especializa en la detección de distintas formas geométricas en imágenes, tales como líneas, círculos, elipses, parábolas, hipérbolas, etc. El problema con este algoritmo es el tiempo y el costo computacional que requiere ya que cada pixel de interés en la imagen es considerado parte de una posible solución y para ello se tiene que evaluar con diferentes valores discretos de un ángulo.

Un algoritmo que es comúnmente usado es el RANSAC (Random sample consensus) el cual es un método iterativo para calcular los parámetros de un modelo matemático de un conjunto de datos que contiene valores atípicos. Este algoritmo no determinístico se caracteriza por aumentar la probabilidad de convergencia a la par que se aumenta el número de iteraciones, el problema del mismo es que RANSAC no siempre es capaz de obtener la mejor solución ya que la inicialización es elegida aleatoriamente. Otra manera de atacar este tipo de problemas es usando metaheurísticas. Este paradigma de algoritmos se ha vuelto más popular estos ultimos años, ya que estas técnicas atacan muy bien a diversos problemas de optimización. Para este caso se utilizó un algoritmo de estimación de distribución llamado UMDA el cual se encarga de encontrar una cierta distribución de probabilidad que genere individuos aptos que maximicen o minimicen la función objetivo, para ello asume independencia en los individuos. Hay que ver que esta técnica utiliza vectores binarios como población para un espacio discreto, y para ello más adeltante se verá cómo se adaptó el problema usando este algoritmo.

Otro enfoque basado en metaheurísticas usado son los algoritmos genéticos, los cuales tratan de imitar el comportamiento natural de la evolución para ir obteniendo nuevos individuos que generen mejores soluciones, esto anterior se logra con dos métodos importantes: **Cruza** que toma dos individuos que generen buenos resultados y en base a ellos se genera un hijo con sus características, **Mutación** que toma un individuo y en base a cierta probabilidad modifica sus genes, esto anterior para mejorar la diversidad del conjunto.

Para el problema actual se tomaron imágenes de arterias que ya habían pasado por un proceso de segmentación y umbralización, y de las cuales se generaron las pruebas.



(a) Figura 1. Imagen procesada de una arteria de retina.

En la Figura 1. podemos ver un ejemplo de imagen con la que se trabajó. Hay que destacar que se tuvo que hacer conversiones de formato PNG a formato PGM, esto para trabajar con las imágenes de una manera más sencilla por lo mismo que este formato es compuesto de una matriz de valores enteros que indican el valor en la escala de grises que representa los pixeles.

### *II-A. Detección de parábolas*

El problema que fue atacado en este trabajo fue el de encontrar un buen modelo de ajuste de parábola a la imagen dada, lo cual implicaba encontrar los coeficientes $A$, $B$ y $C$ tales que hicieran que cumplieran el modelo de la parábola horizontal $Ax^2 + Bx + C = y$ (esto anterior debido a que las mejores parábolas que se encontraban en las imágenes eran horizontales). Para ello, la idea era encontrar tres puntos potenciales (blancos) en la imagen que pudieran ser parte de una parábola y mediante a sus valores $x$, $y$ se generaran los coeficientes usando las siguientes fórmulas:

$$\begin{aligned} A &= \frac{y_k(x_j - x_i) + y_j(x_i - x_k) + y_i(x_k - x_j)}{(y_i - y_j)(y_i - y_k)(y_j - y_k)}, \\ B &= \frac{y_k^2(x_i - x_j) + y_j^2(x_k - x_i) + y_i^2(x_j - xk)}{(y_i - y_j)(y_i - y_k)(y_j - y_k)}, \\ C &= \frac{y_j y_k(y_j - y_k)x_i + y_k y_i(y_k - y_i)x_j + y_i y_j(y_i - y_j)x_k}{(y_i - y_j)(y_i - y_k)(y_j - y_k)} \end{aligned} \tag{1}$$

Dichas fórmulas se pueden deducir al resolver el siguiente sistema de ecuaciones:

$$\begin{bmatrix} y_i^2 & y_i & 1 \\ y_j^2 & y_j & 1 \\ y_k^2 & y_k & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} x_i \\ x_j \\ x_k \end{bmatrix}$$

Con lo anterior, el problema se reduce a la búsqueda de dichos puntos que generan el mejor modelo de parábola para la imagen.

Antes de seguir con el tema de la búsqueda de puntos hay que tomar en cuenta dos cosas. La primera es el dibujo de la parábola, el cual se realizó con un algoritmo llamado: Bresenham de punto medio, el cual toma como entrada el vértice, el parámetro $p$ y una cota en $x$ para el dibujo de la parábola y realiza el dibujo usando la técnica de determinar el pixel más cercano en base a la fórmula de punto medio. Dicho algoritmo ayudó a determinar los valores $x$, $y$ de cada punto de la parábola, los cuales se usarían en el dibujo y la métrica de calidad. Para encontrar el punto del vértice y la distancia focal de la parabola a dibujar podemos utilizar las siguientes fórmulas:

$$\begin{aligned} x_v &= \frac{-B}{2A}, \\ y_v &= C - \frac{B^2}{4A}, \\ 4p &= \frac{1}{A} \end{aligned} \tag{2}$$

El siguiente punto a tomar en cuenta es la métrica de calidad mencionada en el párrafo anterior, dicha métrica fue tomada como la define *(Cruz-Aceves, 2017, p.5)* en el artículo *Fast Parabola Detection Using Estimation of Distribution Algorithms* donde se propone usar el producto de Hadamard para hacer el filtrado de los pixeles de la parábola que se ajustaron a la imagen, a final de cuentas, la función objetivo debe maximizar dicho producto.

Para realizar la búsqueda de los puntos antes mencionados se recurrió al algoritmo UMDA y a un algoritmo genético, los cuales se describirán detalladamente más adelante, pero hay que notar que el planteamiento fue el mismo para ambos algoritmos.

### *II-B. Planteamiento de la búsqueda*

Primeramente se creó una lista con todos los puntos de interés de la imagen. A continuación, como ambos algoritmos se basan en conjuntos poblacionales, se propuso generar individuos codificados en vectores binarios, dichos vectores contenían tres índices binarizados de la lista anterior descrita, con esos índices podíamos obtener puntos aleatorios que en base a las ecuaciones (1) y (2).

Teniendo lo anterior en cuenta hay que definir ahora el tamaño del vector binario de índices, un ejemplo de tamaño dado podía ser de 8 bits por cada índice a considerar, en total se tendría un vector de 24 bits. Para el caso del programa, como el tamaño de la lista de puntos variaba en base a la imagen, se optó por generar un pequeño método que determinara el número de bits a necesitar por índice en el vector, a continuación se muestra el algoritmo:

**Algorithm 1** Tamaño de bits.

```
1: procedure NoBits(listSize)
2:     nbits ← 0.
3:     while listSize > 0 do
4:         nbits ← nbits + 1.
5:         listSize ← listSize / 2.
6:     return nbits
```

Al ejecutar el Algoritmo 1. se calculó la medida en bits por cada individuo del vector binario.

## III. UMDA

El algoritmo de distribución marginal univariada (UMDA) fue la primera opción con la cual se atacaría el problema de la búsqueda de los tres mejores puntos que formarían la parábola. Como se mencionó anteriormente, este algoritmo es una técnica basada en población sin inspiración, se utiliza en el campo de optimización estocástica y está relacionado con los algoritmos genéticos y otras técnicas evolutivas que están inspiradas en la teoría biológica de la evolución y selección natural *(Brownlee, 2015)*.
La estrategia de este algoritmo es formar individuos aptos para maximizar o minimizar la función objetivo del problema que se está atacando, esto anterior en base a una distribución probabilística que el mismo algoritmo va adaptando en base a la población que ya generó.
Para generar la distribución de probabilidad el UMDA trabaja con vectories binarios como se mencionó anteriormente y, en base a ellos trata de aproximar a la distribución $P$ en base a sus marginales:

$$P(x) = \prod_{i=1}^{n} P(X_i = x_i) \tag{3}$$

Para generar dicha aproximación hay que destacar que el algoritmo asume independencia en los individuos de su población, lo cual puede ser algo malo si en el problema que se quiere tratar no existe dicho factor.
Inicialmente el algoritmo genera una población completamente aleatoria de cierto tamaño, posteriormente evalúa dicha población con el método que se adaptó para el problema y ordena a los individuos en base a los resultaron que produjeron. A continuación se realiza un paso elitista donde se conserva en la nueva población el individuo que brindó mejores resultados, y se procede estimar la ditribución de probabilidad con cierta cantidad de individuos que dieron buenos resultados. Esto anterior se repite hasta que se cumpla un cierto número de iteraciones o hasta que se logre convergencia, una vez pase esto podremos encontrar la mejor solución en la parte más alta de la población ya que la hemos estado guardando cada vez que se realiza la evaluación.
Para el caso actual ya se mencionó que los individuos serán los índices de tres pixeles blancos y que la función de evaluación será el generar la parábola con estos tres índices y aplicar el producto de Hadamard para contabilizar el número de pixeles de la parábola que se encontraron en la imagen. De la misma forma se menciona que el criterio de paro fue el cumplir un cierto número de iteraciones, y por lo cual se tuvieron que realizar diferentes pruebas al algoritmo.

**Algorithm 2** Detección de parábola mediante UMDA.

```
1:  procedure UMDA(PopSize, BitsNum, SelectionRate)
2:      Population ← InitializePopulation(BitsNum, PopSize).
3:      EvaluatePopulation(Population).
4:      SBest ← GetBestSolution(Population).
5:      for generation in NoGenerations do
6:          Selected ← SelectFitSolutions(Population, SelectionRate).
7:          V ← CalcFreqOfComponents(Selected).
8:          Offspring ← ∅.
9:          for i to PopSize do
10:             Offspring ← ProbabilisticallyConstructSolution(V).
11:         EvaluatePopulation(Offspring).
12:         SBest ← GetBestSolution(Offspring).
13:         Population ← Offspring.
14:     return SBest
```

En el Algoritmo 2. se puede ver lo que se comentó anteriormente, se genera una población aletaria, se evalua cada individuo, se obtiene la mejor solución y se actualiza la distribución, en el caso para hacer esto último podemos seguir la siguiente fórmula:

$$P_i = \frac{1}{m} \sum_{j=1}^{m} Population_{j,i} \tag{4}$$

Donde $m$ es el radio de selección.

## IV. Algoritmo genético

El otro enfoque que se tomó para atacar el problema de la búsqueda de los índices fue usar un algoritmo genético. Este tipo de algoritmos al igual que el UMDA se utiliza en el área de optimización estocástica y se basan en el proceso de evolución biológica. Dichos algoritmos pueden trabajar con problemas discretos y continuos, y cuentan con dos funciones importantes para encontrar los mejores individuos, las cuales son:

1. **Cruza:** Esta función permite generar nuevos individuos a partir de dos padres que tuvieron un buen desempeño en sus soluciones, esto con el fin de que el hijo herede genes de los mismo padres y mejore las soluciones antes encontradas. Existen en la actualidad muchos métodos de cruza pero para el problema se utilizó el más común llamado *Cruza uniforme* en donde y con cierta probabilidad se elegían ciertos genes (bits) del padre y ciertos genes de la madre para formar el individuo.
2. **Mutación:** Esta operación consiste en tomar un individuo y con cierta probabilidad (muy baja) cambiar algunos de sus genes. Lo anterior permite que siga habiendo diversidad en la población y que no haya convergencia muy rápido a un posible óptimo local. Para el algoritmo usado, se hizo la cruza cambiando ciertos bits del individuo por su complemento.

Estas dos operaciones combinadas con un importante factor de elitismo en la generación de la nueva población permitieron encontrar individuos que dieran buenas soluciones.

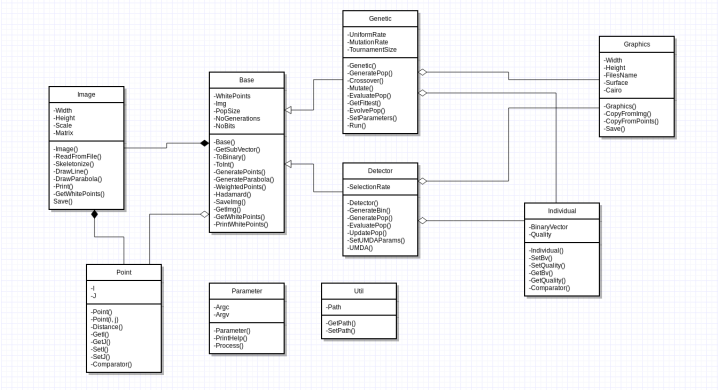
**Algorithm 3** Detección de parábola mediante un algoritmo genético.

**procedure** GENETIC($PopSize, BitsNum, URate, MRate$)
    Population ← InitializePopulation(BitsNum, PopSize).
    SBest ← ∅.
    **for** generation in NoGenerations **do**
        EvaluatePopulation(Population).
        SBest ← GetBestSolution(Population).
        Offspring ← CrossOver(Population, URate).
        Offspring ← Mutation(Offspring, MRate).
        Population ← Offspring.
    **return** $SBest$

Se puede ver que el Algoritmo 3. retorna $SBest$ como el vector que generó mejores resultados. Y los parámetros $URate$ y $MRate$ representan las probabilidades de realizar cruza y mutación.

## V. Arquitectura del sistema

Al buscar un buen diseño de arquitectura para el sistema se decidió utilizar el paradigma de programación orientada a objetos (POO) en el lenguaje C++ ya que permite una buena abstracción de los datos y las entidades entidades, además de una ventaja en la reutilización del código. A continuación se muestra el diagrama UML de las clases usadas con sus atributos y métodos:




(b) Figura 2. Diagrama UML de las clases usadas en el programa.

Primero hay que destacar la clase llamada *Base*, esta se generó porque se detectó una gran similitud en los métodos y atributos que usarían el UMDA y el algoritmo Genético, tales como el tamaño de la población, el número de generaciones, el número de bits por individuo, y métodos como el generador de parábolas y la función de evaluación. Por lo anterior, las clases *Detector* (UMDA) y *Genetic* heredaron de la clase Base con lo que se pudo reutilizar mucho código.
Una clase muy importante usada en todo el programa fue la clase *Image*, esta contenía una matriz de datos que contenía los pixeles de la imagen así como métodos para su tratamiento como el poder leer dicha matriz desde un archivo o el dibujo de líneas y parábolas con el algoritmo de punto medio.
Otras dos clases importantes fueron la de Individuo y la de Gráficos, estas anteriores fueron usadas por los algoritmos de optimización para ir guardando los individuos (vectores binarios) con su respectivo desempeño, así como para generar la imagen de salida que contenía la arteria en conjunto con la mejor parábola encontrada.
Por último hay dos clases que no cuentan con agregaciones o composiciones con las demás, esto porque dichas clases se usaron en la función *main* como auxiliares en ciertas tareas.

## VI. Ejecución, comparación y resultados

Para encontrar los mejores resultados posibles se tuvo que ejecutar los algoritmos varias veces y con diferentes parámetros, tales como la semilla generadora de números aleatorios y los radios de selección cruza y mutación. A cotinuación se muestran los resultados obtenidos por cada algoritmo, así como una comparación de ambos, en donde se muestran algunas imágenes, tablas de tiempos obtenidos y los parámetros usados.

### *VI-A. Resultados UMDA*

En este algoritmo además del tamaño de la población y del número de generaciones se tuvo que variar el radio de selección para obtener diversos resultados. A continuación se mostrarán algunas tablas de los tiempos obtenidos al aplicar el UMDA en un conjunto de 20 imágenes, todas de dimensiones 565x584 pixeles.

Cuadro I: UMDA: Resultados obtenidos con un tamaño de población de 20 individuos y 50 generaciones.

| **Tiempo de ejecución (s).** |
|---|
| 3.93948 s. |
| 3.7443 s. |
| 3.74933 s. |
| 4.32264 s. |
| 4.62275 s. |
| 4.05455 s. |
| 4.09373 s. |
| 4.31462 s. |
| 4.41737 s. |
| 4.11854 s. |
| 3.94432 s. |
| 4.19946 s. |
| 4.03182 s. |
| 3.69991 s. |
| 3.71428 s. |
| 3.59808 s. |
| 3.61779 s. |
| 3.75803 s. |
| 3.8531 s. |

Cuadro II: Medidas estadísticas producidas por los datos de la Tabla 1.

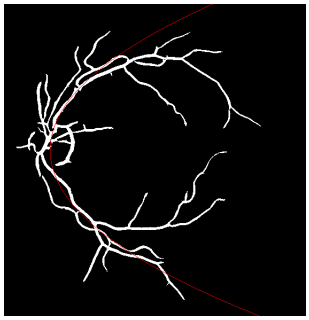
| **Medidas estadísticas** | |
|---|---|
| Media: | 3.9841 s. |
| Máximo: | 4.62275 s. |
| Mínimo: | 3.59808 s. |
| Varianza: | 0.0807 s. |

En las Tablas 1 y 2 podemos ver los resultados de tiempo reportados por el programa a lo largo de la ejecución del

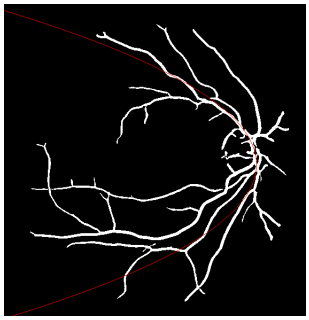
mismo con cada imagen del conjunto de prueba. Podemos notar que en promedio el UMDA tarbada 3.98 segundos en procesar cada imagen con una varianza muy pequeña, esto porque como se comentó anteriormente, el tamaño de las imágenes era el mismo y el algoritmo usó los mismos procedimientos en cada una.

### *VI-B. Imágenes generadas por el UMDA*

A continuación se mostrarán algunas de las imágenes generadas por el algoritmo usando los parámetros anteriormente descritos.

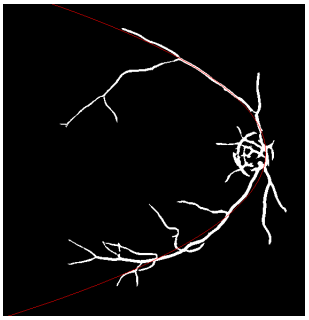


(c) Figura 3. Imagen resultante del UMDA.



(d) Figura 4. Imagen resultante del UMDA.

(e) Figura 5. Imagen resultante del UMDA.



(f) Figura 6. Imagen resultante del UMDA.

En el conjunto anterior de imágenes podemos ver en la mayoría buenos resultados excepto en la Figura 5. donde podemos observar que la parábola se creo en una ramificación derecha donde se puede ver que la parábola se ajustaría mejor a la curva que se abre a la izquierda.

### *VI-C. Resultados algoritmo genético*

Para este algoritmo se realizaron las mismas pruebas que con el UMDA y sobre todo se aplicaron las mismas condiciones, como el tamaño de la población y el número de generaciones. A diferencia del UMDA, este algoritmo requirió de los valores de probabilidad de cruce y mutación, los cuales fueron puestos a: 0.5 y 0.015.

Cuadro III: AG: Resultados obtenidos con un tamaño de población de 20 individuos y 50 generaciones.

| **Tiempos de ejecución (s).** |
|---|
| 3.74618 |
| 3.91177 |
| 4.09252 |
| 4.03615 |
| 4.02431 |
| 4.04685 |
| 3.97711 |
| 4.00251 |
| 4.09495 |
| 4.13126 |
| 4.12499 |
| 4.19146 |
| 4.23933 |
| 3.9498 |
| 4.18817 |
| 4.12121 |
| 3.97177 |
| 4.10107 |
| 4.23086 |

Cuadro IV: Medidas estadísticas producidas por los datos de la Tabla 3.

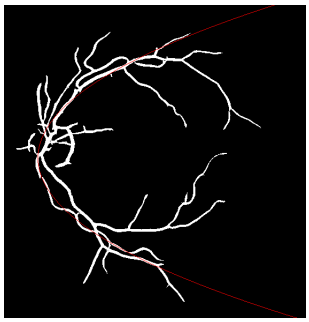
| **Medidas estadísticas** | |
|---|---|
| Media: | 4.0679945 s. |
| Máximo: | 4.23933 s. |
| Mínimo: | 3.74618 s. |
| Varianza: | 0.0145898481 s. |

En las Tablas 2 y 3 podemos apreciar los tiempos de ejecución así como sus medidas estadísticas. Como se mencionó anteriormente, se usaron las mismas condiciones e imágenes que en el UMDA para las pruebas.

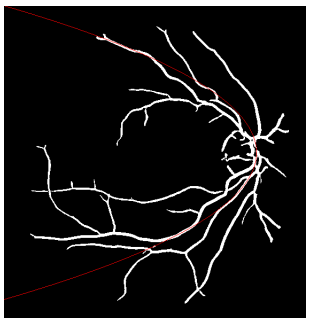
Podemos notar que los tiempos obtenidos son muy parecidos a los del UMDA, aunque este algoritmo tardó en promedio segundos en procesar una imagen, un poco más de tiempo que el UMDA.

### *VI-D. Imágenes generadas por el algoritmo genético.*

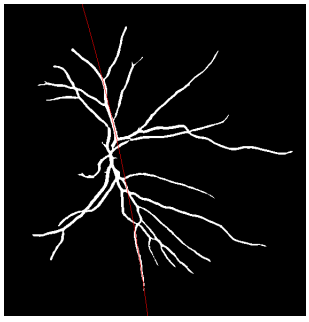
A continuación se muestran las imágenes que el produjo el algoritmo, para hacer una comparación justa, se mostrarán los resultados con las mismas imágenes de entrada que se mostraron en la sección de resultados del UMDA.



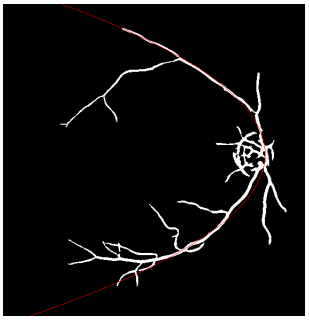
(g) Figura 7. Imagen resultante del AG.



(h) Figura 8. Imagen resultante del AG.



(i) Figura 9. Imagen resultante del AG.



(j) Figura 10. Imagen resultante del AG.

Podemos ver que este algoritmo también nos produjo buenos resultados con las mismas imágenes de entrada, y en el caso de la Figura 9. podemos ver una mejora en la obtención de la parábola a diferencia del UMDA.

## VII. Conclusión

A lo largo de este reporte se ha comentado los algoritmos utilizados para encontrar la mejor parábola en imágenes de arterias de retina. Se pudo ver que el algoritmo genético obtuvo un mejor conjunto de parábolas para el conjunto dado pero en general esto siempre pasó, se hicieron muchas pruebas con diferentes parámetros y se destacó el algoritmo genético por sus resultados. Pese a lo anterior hubo una imagen a la que no se pudo encontrar el mejor modelo de parábola que lo ajustara, esto por la composición de dicha imagen y sobre todo, por la función de evaluación, con dicha función, ambos algoritmos convergían a mínimos locales de forma rápida.

A parte de la función de evaluación anteriormente mencionada se hicieron pruebas con otra función la cual ponderaba a los pixeles que estaban más cerca del vértice y penalizaba a los más alejados, esto para que se produjeran parábolas más cercanas a los centros de las arterias, pero los resultados no fueron muy buenos ya que en la mayoría de las veces la apertura de las mismas no era lo suficientemente buena como para adaptarse a la arteria.

Como mejora en un trabajo futuro se propone trabajar con el algoritmo genético, se puede tratar la población no como vectores binarios, sino como una versión decimal de la misma, a su vez se pueden emplear mejores operadores de cruza y mutación. Otra gran mejora a este algoritmo es convertirlo a un algoritmo memético, en donde se introduce una búsqueda local en lugar del proceso de mutación, esto para explotar las soluciones producidas por el mismo algoritmo.

Otra sugerencia que se propone para un trabajo futuro es probar estos algoritmos con otro tipo de imágenes, tales como imágenes de pies, ya que se pueden hacer buenas adaptaciones a los arcos de los pies con parábolas.

## Referencias


[1] Guerrero-Turrubiates and Cruz-Aceves, *Fast Parabola Detection Using Estimation of Distribution Algorithms*, México: 2017.

[2] Brownlee J. *Clever Algorithms*, USA: 2017.

[3] Talbi E. *Metaheuristics: From Design to Implementation*, USA: 2017.