CORRECCIÓN, ALINEACIÓN Y RECONSTRUCCIÓN 3D DE CORTES SERIADOS: UN CASO DE ESTUDIO

Rhadamés E. Carmona
Universidad Central de Venezuela. Escuela de Computación. Laboratorio de Computación Gráfica.
Los Chaguaramos. Apdo. 47002, Caracas, Venezuela.
email: rcarmona@strix.ciens.ucv.ve
y
Paluszny, Marco
Universidad Central de Venezuela. Escuela de Matemáticas. Laboratorio de Computación Gráfica y Geometría Aplicada.
Los Chaguaramos. Apdo. 47809, Caracas, Venezuela.
email: marco@euler.ciens.ucv.ve

Resumen

En este trabajo se presenta un caso de estudio de reconstrucción 3D, a partir de diez cortes seriados de un parásito, con problemas de iluminación y no alineados. El problema de iluminación es resuelto mediante un método novedoso ajustando cada *pixel* según un polinomio obtenido por mínimos cuadrados sobre los puntos de fondo de cada imagen. La alineación es realizada minimizando el promedio de la suma de los *pixels* de la diferencia entre cada par de imágenes, utilizando Algoritmos Genéticos. Se hace además un estudio de los parámetros del algoritmo en cuanto a la calidad de la alineación y el tiempo de respuesta. La visualización y reconstrucción 3D es realizada mediante la detección y despliegue de contornos de los cortes, utilizando una interfaz gráfica para permitirle al usuario elegir las estructuras internas a desplegar.

Palabras Claves: alineación de imágenes, reconstrucción 3D, visualización 3D, algoritmos genéticos, procesamiento digital de imágenes, corrección de imágenes, computación gráfica.

Abstract

We present a study case of 3D reconstruction of missaligned serial sections of a biological organism, without the support of artificial landmarks. Furthermore, the illumination of the images had to be corrected. The latter consisted in the gray level adjustment of the background pixels using a correcting polynomial function, which determined the right intensity for each pixel in the whole image. Many researchers have worked in the alignment problem, mainly using techniques to minimize various cost functions. We use the average of gray level differences between corresponding pixels in every pair of consecutive images. This function is minimized using a genetic algorithm. For the 3D reconstruction, the system detects contours in the aligned images in an interactive fashion. Thus, the user can select which structures to reconstruct and visualize.

Keywords: images alignment, 3D reconstruction, 3D visualization, genethic algorithms, digital image processing, image enhancement, graphics computing.

1. INTRODUCCIÓN

Para hacer la reconstrucción y visualización 3D de una serie de cortes seriados, es requisito fundamental que los mismos estén alineados. Muchos investigadores han trabajado en el problema de alineación [1,2,5,6,7,8,12,13], utilizando técnicas para minimizar una función que mida la confiabilidad del ajuste. Las soluciones presentes en la bibliografía, pueden resumirse en los siguientes casos [5]:

a. Utilización de marcas artificiales, que indican la alineación correcta.
b. Minimización de la suma de las distancias entre puntos que se corresponden en las dos imágenes.
c. Por inspección visual. Esto significa que interactivamente el usuario superpone dos imágenes con el uso de técnicas de visualización transparente, hasta encontrar el mejor ajuste visual [7].
d. A través de la superposición de los centroides y/o ejes principales de las imágenes [2].
e. Alineación de los contornos de las imágenes [8].
f. Alineación mediante análisis de correlación [2,5,6,8,12].
g. Minimización de la diferencia de los niveles de gris entre las imágenes [1,13].

En cualquier caso, exceptuando el primero, existen contraejemplos en los cuales la alineación generada no es adecuada, porque típicamente este problema está subdeterminado. En el primer caso, cuando se tienen marcas artificiales en las imágenes, el problema es trivial, pero en general estas marcas no están disponibles. En los casos b, d y e, el error puede ser significativo, debido a que se trabaja con una restringida porción de datos de los cortes. En el caso c, por tratarse de un ajuste visual, no se está cuantificando el error, el cual puede propagarse significativamente en la alineación de las imágenes siguientes. En nuestro caso, se minimizará la diferencia entre cada par de cortes consecutivos (caso g), usando un octavo de los pixels de las imágenes, distribuidos uniformemente en filas y columnas.

En muchos casos, el criterio para determinar que se alcanzó la solución puede fallar debido a que en este problema en general existen muchos mínimos locales. Es por ello que se consideran los algoritmos genéticos para minimizar la diferencia entre imágenes consecutivas, ya que en la práctica se ha demostrado que éstos encuentran la solución a problemas de minimización en donde otros métodos fallan.

En este trabajo se cuenta con diez cortes de un parásito. Estos están mal enfocados, presentando una problema en la iluminación, por lo que es necesario un preprocesamiento, que en nuestro caso resultó ser novedoso. Este consistió en determinar factores de corrección en cada pixel a partir de un polinomio obtenido por mínimos cuadrados, sobre los pixels de fondo de una imagen. La reconstrucción 3D es realizada segmentando las imágenes una vez corregidas y alineadas, y haciendo detección de contornos de sólo las estructuras que se desean visualizar en 3D, mediante un programa interactivo realizado en Visual C++, con despliegue OpenGL.

El trabajo está dividido en varias secciones. En la sección 2 se presenta en detalle el preprocesamiento aplicado a los cortes antes de la alineación. La sección 3 trata sobre el algoritmo genético para la alineación de pares consecutivos de imágenes. En la sección 4 se describe el proceso para hacer la reconstrucción y visualización 3D. En la sección 5 se describen los resultados obtenidos, y finalmente se tienen las conclusiones, reconocimientos y bibliografía.

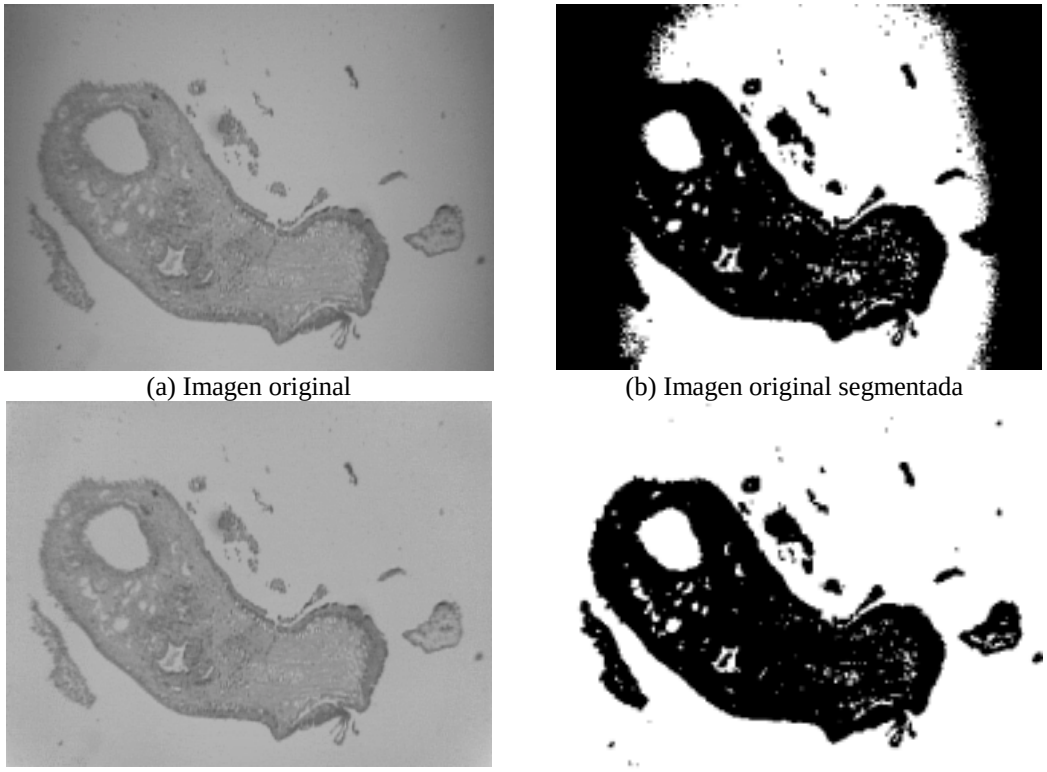
2. PREPROCESAMIENTO DE LOS CORTES

Ocasionalmente las imágenes provenientes de cortes planos de un objeto tienen ruido, que podría causar una incorrecta interpretación de los datos. En nuestro caso, además de que los cortes no están alineados, se presenta un problema de iluminación, posiblemente por fallas del aparato utilizado. En consecuencia, resulta imposible encontrar un umbral que permita segmentar las imágenes adecuadamente. Esto se ejemplifica en la Fig. 2.1(a,b) en la cual se muestra el primer corte de parásito junto a su correspondiente imagen segmentada utilizando el método del umbral[4]. Se nota claramente como la segmentación no separa eficazmente el fondo del resto de la imagen.

La reducción del ruido en una imagen consiste en la homogeneización de su iluminación. Observando las imágenes, vemos que tienen la máxima iluminación cerca del centro, y a mayor cercanía al borde la imagen se va haciendo más obscura de una forma no lineal. Esto sugirió ajustar por mínimos cuadrados los puntos de fondo de la imagen por un polinomio que en nuestro caso bastó ser de grado 2. Con este polinomio $P(x,y)$ se puede corregir la iluminación en cada pixel $A[x_i,y_i]$ de la imagen de ancho *Width* y largo *Height* multiplicando el pixel por un factor en función de $P(x_i,y_i)$. Esta corrección se puede modelar con la siguiente fórmula:

$$A_c[x_i, y_i] = A[x_i, y_i] * \frac{Max(P(x, y))}{P(x_i, y_i)} \qquad (\&2.1)$$

en donde $Max(P(x,y))$ es el valor máximo del polinomio en el dominio $[0,Width-1]_x[0,Height-1]$, y $A_c[x,y]$ es la imagen corregida. Dado que el valor $Max(P(x,y))$ se encuentra cerca del punto de máxima iluminación, la corrección en los pixels cercanos a dicho punto es pequeña, mientras que la corrección crece a medida que el pixel está más lejano.



(a) Imagen original (b) Imagen original segmentada

(c) Imagen corregida (d) Imagen corregida segmentada

Figura 2.1: Corrección de iluminación y segmentación de un corte.

El método utilizado para encontrar los coeficientes del polinomio $P(x,y)$ que mejor ajusta a los puntos del fondo de la imagen, es mínimos cuadrados.

$$Min \sum_{(x_k, y_k)} (A[x_k, y_k] - P(x_k, y_k))^2 \qquad (\&2.2)$$

En nuestro caso, bastó con tomar sólo 100 puntos del fondo de la imagen (aprox. 0.1% del total de puntos de las imágenes de resolución 320x240) para realizar la minimización, ya que en la práctica utilizar más puntos no afectaba significativamente en la solución.

Denotemos el polinomio como $P(x, y) = b_0 x^2 + b_1 xy + b_2 y^2 + b_3 x + b_4 y + b_5$. Debido a que deseamos minimizar la ecuación &2.2, derivamos respecto a cada b_w (w=0,...,5) e igualamos a cero, obteniendo en total 6 ecuaciones de la forma:

$$\sum_{(x_k, y_k)} P(x_k, y_k) . \frac{\partial P(x_k, y_k)}{\partial b_w} = \sum_{(x_k, y_k)} A(x_k, y_k) . \frac{\partial P(x_k, y_k)}{\partial b_w} \qquad (\&2.3)$$

Con estas 6 ecuaciones formamos un sistema $M.b=c$, donde M es una matriz simétrica de dimensiones 6x6, $b=(b_0, ..., b_5)^t$ el vector columna solución, y c el vector independiente formado por el lado derecho de la igualdad en las ecuaciones

&2.3. Para resolver el sistema de ecuaciones, pueden utilizarse diversos métodos por las características de la matriz. Sin embargo, se utilizó el método de eliminación Gaussiana con pivoteo total, seguido de una sustitución hacia atrás para obtener el vector solución. Cabe destacar que no es conveniente resolver el sistema directamente con la numeración original de los pixels, ya que se generan valores inmanejables. Por esto se realiza previamente una normalización de los datos, que consiste en aplicar el dominio de la imagen $[0, Width-1]x[0, Height-1]$ en $[0,1]x[0,1]$, e igualmente el rango (tonos de grises) de $[0,255]$ a $[0,1]$.

Una vez determinado $P(x, y)$ en el dominio normalizado de *A*, se hace el ajuste de los puntos (x_k,y_k) de la imagen en el dominio original mediante:

$$A_c(x_k, y_k) = A(x_k, y_k) * \frac{Max\{P(x, y) / 0 \le x, y \le 1\}}{P\left(\frac{x_k}{Width-1}, \frac{y_k}{Height-1}\right)}$$

En la Fig. 2.1(c,d) vemos la imagen corregida y su segmentación por el método del umbral, aplicado al primer corte del parásito. Como el error de iluminación visualmente es similar en todos los cortes del parásito, se utilizó el mismo polinomio obtenido de la primera imagen para ajustar todas las demás.

3.- ALINEACIÓN DE IMÁGENES

La alineación de dos imágenes *A* y *B* consiste en encontrar la traslación (*Tx*, *Ty*) y el ángulo de rotación α alrededor de un punto fijo (*Cx*, *Cy*) en la imagen *B* que optimicen alguna función que mida la confiabilidad del ajuste. En principio, la función usada para medir la confiabilidad es la diferencia entre las dos imágenes $f(\alpha, Tx, Ty) = B' - A$, donde *B*' es la imagen *B* transformada. A menor diferencia, mejor es la calidad de la alineación. El centro de rotación fue fijado como el centro de la imagen, i.e. *Cx*=*Width*/2 y *Cy*=*Height*/2.

El problema de alinear *NC* imágenes A_1, A_2, ..., A_{NC} se reduce a la alineación de pares consecutivos A_k, A_{k+1}; esto es, determinar los parámetros *Tx*, *Ty* y α para cada par, y luego realizar la composición de las transformaciones para obtener las imágenes deseadas.

La alineación de un par A_k, A_{k+1}, que denotaremos *A* y *B*, es hecha mediante un algoritmo genético con las siguientes características:

- Función objetivo: $Min\ f(\alpha, Tx, Ty)$.
- Mecanismo de selección: Rueda de la Ruleta sobre la función de adaptación renormalizada.
- Operador de cruce a un punto.
- Operador de mutación a un punto.
- Esquema de sustitución: Elitismo (sobrevive el individuo más apto) y GAP Generacional (sobreviven los *r* individuos más aptos).
- Renormalización lineal.

Definición de la Función Objetivo

Para cuantificar la diferencia de dos imágenes mediante un sólo número, sumamos el módulo de los pixels de la imagen *C*=*B*'−*A*.

$$f(\alpha, Tx, Ty) = \sum_{(x',y') \in B'} \left| B'(x', y') - A(x', y') \right|$$

Sean [*x*,*y*] índices enteros del pixel *B*[*x*,*y*]. Al rotar [*x*,*y*] α° alrededor de (*Cx*, *Cy*) y trasladarlo en (*Tx*,*Ty*), obtenemos un punto (*x*',*y*'), en donde *x*',*y*' no necesariamente son enteros. La intensidad *B*[*x*,*y*]=*B*(x',y') es conocida; en cambio, suponiendo que (*x*',*y*') se encuentre dentro del dominio de *A*, *x*' y *y*' no necesariamente son enteros, por lo que hay que

especificar cómo se determina el valor en $A(x',y')$. En este último caso tenemos un sobremuestreo (*Supersampling*), porque se hace muestreo sobre una imagen digitalizada[3]. El sobremuestreo puede hacerse de varias maneras. En este trabajo utilizamos dos enfoques, lo cual nos define dos funciones objetivo distintas: f_1 (tomando el centro de pixel más cercano) y f_2 (utilizando interpolación bilineal).

Para los puntos (x',y') que no están dentro del dominio de A, el valor $A(x',y')$ no está definido, trayendo como consecuencia que, para distintas transformaciones, la cantidad de puntos de B' que tienen imagen en A puede variar drásticamente. Por ello, se toma $f(\alpha,Tx,Ty)$ como el promedio entre los r valores $|B'(x',y')-A(x',y')|$ para los cuales (x',y') si está dentro del dominio de A.

$$f(\alpha,Tx,Ty)=\frac{1}{r}\sum_{(x',y')\in A'}\left|B'(x',y')-A(x',y')\right|$$

Centro de pixel más cercano

Este consiste en sustituir $A(x',y')$ por $A[u,v]$, en donde $[u,v]$ es el centro del único pixel que contiene a (x',y'). En la bibliografía, esto es llamado Muestreo puntual [3], con la diferencia de que la señal muestreada es discreta.

Interpolación bilineal

En el caso anterior se está discriminando que tan cerca o lejos del centro del pixel $[u,v]$ está el punto (x',y'). Esto crea un problema denominado *aliasing* en el muestreo[11]. Este problema es reducido utilizando interpolación bilineal, el cual puede implementarse eficientemente con sólo tres mutiplicaciones: $A(x',y') = A(x',v)+t_*\{A(x',v+1)-A(x',v)\}$, donde

$A(x',v) = A[u-1,v]+w_*\{A[u,v]-A[u-1,v]\}$,
$A(x',v+1) = A[u-1,v+1]+w_*\{A[u,v+1]-A[u-1,v+1]\}$, con $t = y'-v$ y $w = x'-(u-1)$.

Algoritmo incremental para evaluar la función objetivo

En los dos tipos de funciones objetivos que se especificaron, algorítmicamente hay un conjunto de operaciones que hay que realizar, en donde cabe resaltar el procedimiento Hallar(x',y'), esto es, aplicar la transformación de rotación y traslación al punto (x,y), con $0 \le x \le Width-1$, y $0 \le y \le Height-1$. A alto nivel, el algoritmo para evaluar la función a minimizar puede escribirse como:

```
Función Evaluar(A,B: Imagen; Tx,Ty,α: Real) : Real
 Var x, y: real; i, j, r,S: Entero
 r := 0
 S := 0
 Para x:=0 hasta Width-1 hacer
      Para y:=0 hasta Height-1 hacer
            Hallar (x', y')
            Si (x',y')∈A Entonces
                  S := S+Absoluto(B[x,y]-A(x',y'))
                  r := r+1
            FinSi
      FinPara
 FinPara
 := S/r
FinFunción
```

El procedimiento Hallar(x',y') es costoso si no se toman consideraciones de eficiencia para evitar cálculo redundante. En principio, esta acción equivale a realizar:

```
x' := x*cos(α)-y*sen(α)+0.5*Cx*(1-Cos(α))+0.5*Cy*sen(α)+Tx
```

```
y' := x*sen(α)+y*cos(α)+0.5*Cy*(1-Cos(α))-0.5*Cx*cos(α)+Ty
```

Podemos notar que $0.5*Cx*(1-\cos(\alpha))+0.5*Cy*\text{sen}(\alpha)+Tx$ puede ser calculada una sóla vez, al igual que $0.5*Cy*(1-\cos(\alpha))-0.5*Cx*\cos(\alpha)+Ty$, ya que no dependen de (x,y). Similarmente no debe calcularse $\text{sen}(\alpha)$ ni $\cos(\alpha)$ en cada iteración. Además, se considera que:

- Por cada incremento de x, x' se incrementa en $\cos(\alpha)$, y y' en $\text{sen}(\alpha)$.
- Por cada incremento de y, x' se incrementa en $-\text{sen}(\alpha)$, y y' en $\cos(\alpha)$.

Todas estas consideraciones transforman el algoritmo en un procedimiento incremental con aritmética real. A continuación se muestra el algoritmo.

```
Función Evaluar(A,B: Imagen; Tx,Ty,α: Real) : Real
  Var x,y,x',y',fisrtY,seno,coseno: real; r,S: Entero
  r := 0; S := 0
  seno := sen(α)
  coseno := cos(α)
  x' := 0.5*(Cx*(1-coseno)+Cy*seno)+Tx
  firstY := 0.5*(Cy*(1-coseno)-Cx*seno)+Ty
  Para x:=0 hasta Width-1 hacer
       y' := firstY
       Para y:=0 hasta Height-1 hacer
            Si (x',y')∈A Entonces
                 S := S+Absoluto(B[x,y]-A(x',y'))
                 r := r+1
            FinSi
            y' := y'+coseno
            x' := x'-seno
       FinPara
       x' := x'+coseno
       firstY := firstY+seno
  FinPara
  := S/r
FinFunción
```

4. VISUALIZACIÓN Y RECONSTRUCCIÓN 3D

La reconstrucción 3D del objeto en estudio es realizada colocando en planos paralelos ciertas estructuras que el usuario desee visualizar. Para ello se hizo una interfaz en donde se puede especificar qué contornos visualizar de cada corte. Para la detección de contornos se utilizó el algoritmo de seguimiento de Fronteras[4] (*"Border Following"* o sencillamente BF), y para ello el programa trabaja internamente con las imágenes segmentadas mediante el método del umbral, aunque son desplegadas las imágenes en tonos de grises.

El algoritmo BF tiene como objetivo encontrar todos los puntos de borde de un subconjunto S en una imagen binaria. Para ello se necesita hallar un par inicial de pixels adyacentes In_0 y Out_0, en donde $In_0 \in S$ y $Out_0 \in S^c$. No es fácil para el usuario especificar estos dos puntos explícitamente. Por ello se le permite al usuario utilizar el botón izquierdo del ratón para señalar un punto P cercano al contorno que desea detectar. Algorítmicamente se buscan los puntos adyacentes In_0 y Out_0 más cercanos a P en ocho direcciones, tal y como se muestra en la Fig. 4.1.

Para desplegar los contornos en 3D se utilizó la herramienta gráfica OpenGL, permitiendo las operaciones básicas de acercar/alejar y de rotar respecto a los tres ejes de coordenadas. El programa consta de cinco vistas: tres vistas correspondientes a tres cortes consecutivos, una vista OpenGL, y una vista que muestra los nombres de las imágenes (ver Fig.4.2). El usuario puede definir en esta última, qué cortes desea visualizar en las vistas de cortes. En las tres vistas de cortes, el usuario puede definir cuáles estructuras del parásito visualizar, bastando con presionar el ratón dentro de los contornos correspondientes.

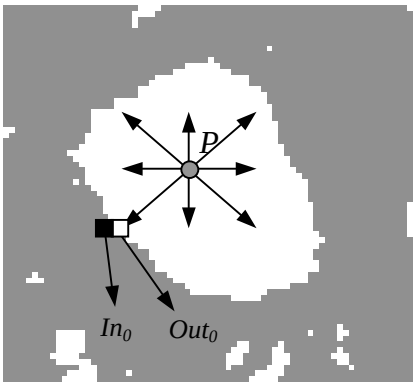


Figura 4.1: Búsqueda de In_0 y Out_0 a partir del punto P especificado por el usuario.

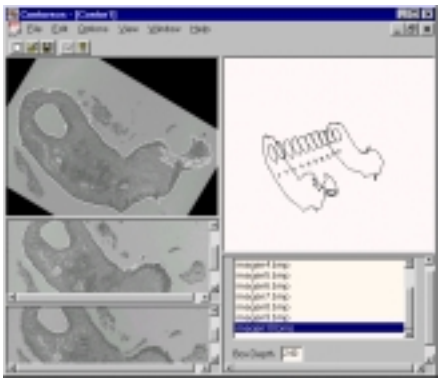


Figura 4.2: Reconstrucción y visualización 3D del parásito

5. RESULTADOS

Para el algoritmo genético, se utilizó un generador de números aleatorios conocido (*rand3*) propuesto por Knuth, el cual está basado en el método Congruencial y en un método de Substracción[11]. El algoritmo genético fue implementado en Visual C++ con procesador Pentium II de 300 MHz. Por eficiencia, para evaluar la función objetivo del algoritmo genético, se tomó aproximadamente $^{100}/_{8}$ % de los puntos (x,y) de B, i.e. los (x,y) tales que x y y son múltiplos de 8. Los parámetros de alineación obtenidos de esta manera son prácticamente los mismos, y el tiempo referente a la evaluación de f es aproximadamente un octavo del tiempo original.

Debido a que las imágenes 6 y 7 eran las que estaban más desalineadas, los análisis que se presentan corresponden a la alineación de éstas. Se hizo una primera prueba para determinar los parámetros de probabilidad de cruce y mutación más adecuados. Esta consistió en realizar 100 corridas de 200 generaciones, con el esquema de sustitución elitesco. El tiempo para una corrida de 200 generaciones con sustitución elitesca fue en promedio de 14 segundos utilizando f_1, y 30 segundos utilizando f_2. La mejor combinación de parámetros fue de 0.04 como probabilidad de mutación (f_1 y f_2), y como probabilidad de cruce 0.8 usando f_1, y 0.85 usado f_2. Note que la probabilidad de mutación adecuada resultó ser relativamente alta, debido a que la función objetivo tiene muchos mínimos locales.

Con los parámetros obtenidos de la primera prueba, la siguiente consistió en determinar la función objetivo más conveniente estudiando los resultados para las variables Tx, Ty y α en 100 corridas. También fue considerado el valor de la función y el tiempo de respuesta.

Para hacer un estudio de las soluciones, se halló previamente la solución óptima para cada función objetivo, mediante una búsqueda exhaustiva (método enumerativo), y no con algoritmos genéticos. Esta solución puede verse en la tabla 5.1. Se puede notar que la diferencia en los valores de Tx_{opt}, Ty_{opt}, α_{opt} y f_{opt} obtenidos usando f_1 y f_2 no es muy significativa.

Función/resultado	Tx_{opt}	Ty_{opt}	α_{opt}	$f_{opt}(Tx_{opt}, Ty_{opt}, \alpha_{opt})$
f_1	-5.52	23.11	0.6700	15.66
f_2	-6.32	22.48	0.6708	15.82

Tabla 5.1: Solución óptima en la alineación de las imágenes 6 y 7

En la Tabla 5.2 se muestra el promedio en las 100 corridas de Tx, Ty α, y función objetivo, utilizando f_1 y f_2. Se muestra además, en promedio, el error que se comete en cada variable. Note que el error promedio en la traslación Tx, Ty no supera 0.06 pixels en cualquier caso, y el error en α es inferior a 0.005 radianes (0.28º).

Item \ Casos	Utilizando f_1	Utilizando f_2
Tx_{prom}	-5.57	-6.31
Ty_{prom}	23.05	22.45
α_{prom}	0.6706	0.6711
f_{prom}	15.76	15.83
$Error(Tx)=\|Tx_{prom}-Tx_{opt}\|$	0.05	0.01
$Error(Tx)=\|Ty_{prom}-Ty_{opt}\|$	0.06	0.02
$Error(\alpha) =\|\alpha_{prom}-\alpha_{opt}\|$	0.0006	0.0033
$Error(f) =\|f_{prom}-f_{opt}\|$	0.10	0.01
$Tiempo\ promedio\ (seg.)$	14.16	30.19

Tabla 5.8: Promedio de *Tx*, *Ty*,α y f_i en las 100 corridas

En general los errores para *Tx*, *Ty*, *f*, son menores utilizando f_2. Esto tiene sentido debido a que mediante f_2 se hace "*antialiasing*" en el muestreo de cada pixel, lo cual puede suavizar la función objetivo, y por lo tanto mejorar la convergencia del AG. Se puede pensar que f_2 es la función más adecuada. Sin embargo, el tiempo empleado para evaluar esta función es mayor (el doble aproximadamente) que utilizando f_1, por requerir interpolación bilineal al evaluar $A(x',y')$. Así que podemos dejar libre al usuario de escoger entre mejor calidad en la solución (en cuyo caso utilizar f_2) o menor tiempo de respuesta (en cuyo caso utilizar f_1). En la tabla 5.3 mostramos el rango de valores resultantes en cada variable en el 99% de las corridas. Los resultados de la peor corrida no fueron tomados en cuenta, por ser atípicos.

Variable/Corridas	Usando f_1	Usado f_2
Tx	[-6.19,-5.42]	[-6.46,-6.19]
Ty	[22.27,23.29]	[22.10,22.93]
α	[0.6596, 0.6768]	[0.6703, 0.6850]

Tabla 5.3: Rango de las variables en 99/100 corridas

Para ambas funciones, el rango de las variables para *Tx*, *Ty*, y α es estrecho, lo cual es un indicativo claro de la convergencia del algoritmo hacia la solución en el 99% de las corridas, con un aceptable margen de error. Comparando la tabla 5.1 con la 5.3 notamos que la solución óptima yace dentro de cada intervalo para ambos casos. El error máximo cometido en alguna corrida usando f_1 es de menos de un pixel para *Tx* y *Ty*, y de 0.01 radianes en el ángulo, mientras que usando f_2 es menos de ½ pixel para *Tx* y *Ty*, y 0.01 radianes en el ángulo. Claramente con f_2 la solución genera menor error, motivado a que la evaluación de la función objetivo es más precisa por utilizar interpolación bilineal.

La tercera prueba consistió en comparar Elitismo y GAP Generacional (r=20), utilizando f_1 y f_2, con los parámetros de mutación y cruce obtenidos de la primera prueba. Usando GAP Generacional los tiempos de respuesta disminuyeron casi en proporción 1 a 5, y los resultados fueron prácticamente iguales, si se eliminan los 10 peores resultados. Esto se debió a que con GAP Generacional es más probable que el algoritmo genético converja hacia un mínimo local, por lo que el 10% de las corridas aportaron una solución lejana a la deseada.

6. CONCLUSIONES

Se propuso un método efectivo y novedoso para corregir el error de iluminación de los cortes del parásito.

En un tiempo prudencial, los algoritmos genéticos pueden resolver el problema de alineación de imágenes. Utilizando Elitismo, el tiempo de respuesta promedio de una corrida de 200 iteraciones fue de 14.15 segundos usando f_1, y 30.19 segundos usando f_2. Utilizando GAP Generacional, y sustituyendo el 20% de la población (r=20), el tiempo de respuesta para una corrida fue aproximadamente 1/5 del tiempo con Elitismo, i.e. 3 segundos usando f_1 y 6 segundos usando f_2. Sin embargo, usando Elitismo, sólo en 1/100 corridas hubo convergencia a un mínimo local, mientras que utilizando GAP Generacional, fueron 10/100. De esta manera, si deseamos obtener una solución aceptable con pocas corridas (por ejemplo cinco), es recomendable utilizar Elitismo, mientras que si se desean hacer más corridas sin aumentar considerablemente el tiempo total de respuesta, es recomendable utilizar GAP Generacional.

El uso de la función objetivo f_2 (interpolación bilineal) genera resultados de mejor calidad que usando f_1 (centro de pixel más cercano) debido la implementación del *antialiasing* en el primer caso. Sin embargo, debido al filtro aplicado con f_2, el tiempo de respuesta es aproximadamente el doble que al utilizar f_1. Así, desde el punto de vista del usuario del sistema, se podría decidir entre calidad y tiempo de respuesta a la hora de seleccionar una de estas opciones.

Como recomendación, se puede mejorar la visualización 3D de los contornos, eliminando parcialmente líneas ocultas mediante el algoritmo de *Haloed Lines* [3].

7. RECONOCIMIENTOS

Los autores están agradecidos al Dr. H. Arrechedera de la Facultad de Medicina en la UCV por proveer las imágenes del organismo biológico. También estamos agradecidos por fructíferas discusiones.

REFERENCIAS

1. Andreasen A., Drewes A.M., Assentoff J.E. and Larsen N.E. "*Computer-assisted Alignment of Standard Serial Sections without use of Artificial Landmarks. A Practical Approach to the Utilization of Incomplete Information in 3-D Reconstruction of the Hippocampal Region*". Journal of Neuroscience Methods, No. 45, 1992, pp. 199-207.

2. Banerjee P. K., Toga A. W. "*Image Alignment by Integrated Rotational and Translational Transformation Matrix*". Medical Physics, Vol. 39, 1994, pp. 1969-1988.

3. Foley J., van Dam Andries, Feiner S., Hughes J. "*Computer Graphics: Principles and Practice*". Addison-Wesley. Segunda edición. 1990.

4. Galíndez V. "*Herramienta de Ayuda para la Construcción Automática del Careotipo*". Trabajo Especial de Grado. Biblioteca Alonso Gamero. Universidad Central de Venezuela. 1996.

5. Hibbard L. S., Grothe R.A., Jr, Arnicar-Sulze T.L, Dovey-Hartman B. J., Page R.B. "*Computed Three-dimensional Reconstruction of Median-eminence Capillary Modules: Image Alignment and Correlation*". Journal of Microscopy, Vol. 171, Pt 1, July 1993, pp. 39-56.

6. Hristov D.H., Fallone B.G. *"A Gray-level Image Alignment Algorithm for Registration of Portal Images and Digitally Reconstructed Radiographs*". Medical Physics, Vol. 23, No. 1, Enero 1994, pp. 75-84.

7. Isam H. Habboush, Karl D. Mitchell, Robert V. Mulkern, Patrick D. Barnes, S. Ted Treves. "*Registration and Alignment of Three-dimensional Images: An Interactive Approach*". Radiology, Vol. 199, Number 2. Mayo 1996, pp. 573-578.

8. McParland B. J., Kumaradas J. C. "*Digital Portal Image Registration by Sequential Anatomical Matchpoint and Image Correlations for Real-time Field Alignment Verification*". Medical Physics, Vol. 22, No. 7, Julio 1995, pp. 1063-1075.

9. Montgomery K., Ross M. "*Improvements in Semiautomated Serial-section Reconstruction and Visualization of Neural Tissue from TEM Images*", (San José, CA: SPIE, 1993), 69.

10. Montgomery K., Ross M. "*Non-fiducial, Shape-based Registration of Biological Tissue*". NASA Ames Research Center, Moffett Field, CA 94035.

11. Press W., Teukolsky S., Vetterling W., Flannery B. "*Numerical Recipes in C*". Segunda edición. Cambridge University Press. 1992.

12. Radcliffe T., Rajapakshe R., Shalev S. "*Pseudocorrelation: A Fast, Robust, Absolute, Grey-level Image Alignment Algorithm*". Medical Physics, Vol. 21, No. 6, Junio 1994, pp. 761-769.

13. Slomka P. J., Hurwitz G. A., Stephenson J., Cradduck T. "*Automated Alignment and Sizing of Myocardial Stress and Rest Scans to Three-Dimensional Normal Templates Using an Image Registration Algorithm*". The Journal of Nuclear Medicine. Vol. 36, No. 6, Junio 1995, pp. 1115-1122.