Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Maestría en Ciencias en Ciencias de la Computación

Optimización del flujo óptico usando programación genética multi-árbol

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:

Héctor Cepeda Juárez

Ensenada, Baja California, México

2017

Tesis defendida por

Héctor Cepeda Juárez

y aprobada por el siguiente Comité

Dr. Gustavo Olague Caballero

Codirector del Comité

Dr. Pedro Gilberto López Mariscal

Codirector del Comité

Dr. Ubaldo Ruíz López

Dra. María del Carmen Maya Sanchéz

Dra. Carmen Guadalupe Paniagua Chávez

Dr. Jesús Favela Vara

Coordinador del Programa de Posgrado en Ciencias de la Computación

Dra. Rufina Hernández Martínez

Directora de Estudios de Posgrado

Resumen de la tesis que presenta **Héctor Cepeda Juárez** como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación

Optimización del flujo óptico usando programación genética multi-árbol

Resumen aprobado por:

Dr. Gustavo Olague Caballero	Dr. Pedro Gilberto López Mariscal
Codirector de Tesis	Codirector de Tesis

La visión por computadora es una rama de la inteligencia artificial, que tiene por objetivo modelar matemáticamente los procesos de percepción visual en los seres vivos y generar programas que permitan simular estas capacidades visuales en una computadora. La estimación del flujo óptico es uno de los problemas en la visión por computadora que lleva estudiandose desde hace varias décadas, se origina por el cambio en los patrones de la intensidad en la imagen debido al movimiento aparente de los objetos que se encuentran en la escena. Considerando $I(x, y, t)$ como la intensidad de la imagen en un punto(x, y) en un tiempo t, el objetivo es estimar un vector (u, v) para cada punto en la imagen, tal que $I(x, y, t)$ y $I(x + u, y + v, t + 1)$ sean correspondientes. En este trabajo de tesis se describe una técnica basada en la programación genética, que es capaz de construir operadores de manera automática para optimizar el flujo óptico de una secuencia de imágenes, con la finalidad de mejorar la precisión en la estimación y así obtener resultados competitivos comparado con métodos propuestos por expertos. Para medir la precisión del flujo óptico obtenido, se emplea el conjunto de secuencias de imágenes de la base de datos de Middlebury para flujo óptico.

Palabras Clave: **programación genética, visión por computadora, flujo óptico.**

Abstract of the thesis presented by **Héctor Cepeda Juárez** as a partial requirement to obtain the Master of Science degree in Master in Computer Science in Computer science.

Optical flow optimization using multi-tree genetic programming

Abstract approved by:

Dr. Gustavo Olague Caballero	Dr. Pedro Gilberto López Mariscal
Thesis Co-Director	Thesis Co-Director

Computer vision is a branch of Artificial Intelligence with the aim of generating computer programs that imitate the human visual process. Optical flow is a computer vision problem that originates from the change of pattern intensities on the image due to the apparent motion of objects in the scene. We can define $I(x, y, t)$ as the intensity function of image at a point (x, y) at time t, the goal is estimate a vector (u, v) for each point in the image such that $I(x, y, t)$ and $I(x+u, y+v, t+1)$ are corresponding. This document describes a technique based on genetic programming which is able to construct automatically operators to optimize the optical flow of a sequence of images, in order to improve the precision in the estimation and thus obtaining competitive results compared with methods proposed by experts. In order to measure the precision of the optical flow, we use the set of image sequences from the Middlebury database for optical flow.

Keywords: **genetic programming, computer vision, optical flow**

Dedicatoria

Con cariño, a mis padres

Agradecimientos

A mi familia, en especial a mis padres, que siempre me han impulsado a seguir adelante, por creer en mí y ser parte fundamental en el cumplimiento de mis metas, por estar siempre ahí dándome consejos, una llamada de atención cuando la merecia y unas palabras de aliento cuando más las he necesitado. Estaré eternamente agradecido con ustedes.

Agradezco a mis asesores, el Dr. Gustavo Olague y el Dr. Gilberto López por su paciencia y por haberme guiado a la culminación de esta tesis, así como a los miembros de mi comité por sus comentarios y aportes para este trabajo.

Gracias a mis compañeros del posgrado y amigos Mariana, Juan, Selene, Jessica y Luis, por su amistad y por haber hecho más amena mi estadía en estos años de estudio.

A mis amigos de hace ya varios años que se mantuvieron al pendiente de mí durante el tiempo que pasé en la ciudad de Ensenada, en especial a Jonathan, Néstor, Javier, Gloria y Araceli, que de una u otra manera me han brindado su apoyo y buena vibra, se los agradezco.

Al Centro de Investigación Científica y de Educación Superior de Ensenada.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de maestría.

Tabla de contenido

Página

Lista de figuras

Figura Página

Figura Página

Figura Página

Lista de tablas

Capítulo 1. Introducción

La visión por computadora es una rama de la inteligencia artificial, que tiene por objetivo modelar matemáticamente los procesos de percepción visual en los seres vivos y generar programas que permitan simular estas capacidades visuales en una computadora. La estimación del flujo óptico es uno de los problemas que pertenece a la visión por computadora que más se ha estudiado en los ultimos años así; nuevos métodos se han introducido y algunos otros más se han mejorado. Dentro de las aplicaciónes que tiene y no limitandosé a éstas podemos encontrar seguimiento de objetos, detección de movimiento , segmentación de objetos.

El flujo óptico se puede definir como el campo de velocidad en 2D que se origina debido al movimiento aparente de los objetos dentro de una secuencia de imágenes. Considerando $I(x, y, t)$ como la intensidad de la imagen en un punto(x, y) en un tiempo t, el objetivo es estimar un vector de velocidad (u, v) para cada punto en la imagen, tal que $I(x, y, t)$ y $I(x + u, y + v, t + 1)$ sean correspondientes.

Dentro de los métodos que se encuentran en la literatura para realizar la estimación del flujo óptico destaca el propuesto por (Horn, 1981), en el cual introdujo una restricción global para el campo de velocidad en la que se asume que el flujo varía de manera suave. Este método ha servido como base para otros más que se han venido desarrollando y perfeccionando a lo largo del tiempo y más adelante se abordará con más detalle.

A pesar de la diversidad de los métodos existentes y la precisión con la que estos estiman el flujo óptico, hoy en día siguen presentandose ciertas dificultades al momento de realizar la estimación, como lo son las discontinuidades de movimiento debido al solapamiento entre objetos, desplazamientos largos, cambios en la iluminación de la escena así como los costos computacionales, lo que lo hace un problema que sigue abierto para la propuesta de nuevos métodos que realicen de manera más precisa y rápida la estimación de los vectores de flujo.

1.1. Objetivo General

Desarrollar un método basado en la programación genética que sea capaz de construir operadores de manera automatizada que sean aplicables al flujo óptico, con la finalidad de obtener una

precisión en la estimación mejor a la que es obtenida por métodos propuestos por expertos. Para medir el rendimiento de la estimación se usará el conjunto de secuencias de la base de datos de Middlebury para flujo óptico.

1.2. Objetivos Específicos

- Analizar algunos de los métodos que realicen la estimación del flujo óptico que se encuentren en la literatura.
- Proponer un método para realizar la estimación del flujo óptico bajo la estrategia de programación genética.
- Realizar la implementación del método en Matlab.
- Evaluar los resultados obtenidos con la implementación.

1.3. Organización del documento

En el Capítulo 2 se revisa los conceptos de la estimación del flujo óptico, se abordan los métodos propuestos por Horn-Schunck así como el de Lucas-Kanade. También se ven los tipos de representación visual para el flujo óptico, así como las métricas para evaluar la precisión en su estimación.

El Capítulo 3 trata sobre programación genética, que es una técnica de computo evolutivo usada en el desarrollo del presente trabajo de tesis.

El Capítulo 4 trata del método propuesto para mejorar la precisión en la estimación del flujo óptico. Se describen los detalles de la implementación asi como algunas técnicas de procesamiento de imágenes utilizadas.

En el Capítulo 5 se presentan los experimentos realizados con los respectivos resultados obtenidos. Finalmente en el Capítulo 6 se exponen las concluciónes y trabajo futuro.

Capítulo 2. Marco teórico

2.1. Introducción

La estimación del Flujo óptico es uno de los problemas en la visión por computadora que lleva estudiandose desde hace varias décadas. El fenómeno estudiado se origina por el cambio en los patrones de la intensidad en la imagen debido a el movimiento aparente de los objetos que se encuentran en la escena o bien del movimiento del observador.

Este término fue introducido por primera vez por el psicólogo James J. Gibson, mientras realizaba investigación en psicología de la aviación durante la segunda guerra mundial. En su trabajo (Gibson, 1950) habla sobre información percibida por un observador y como esta es usada para guiar su movimiento dentro de su entorno, así descubrió que los cambios en lo que el denominó como ”patrones de flujo óptico”, proveen de información al observador acerca de que tipo de movimiento se esta realizando.

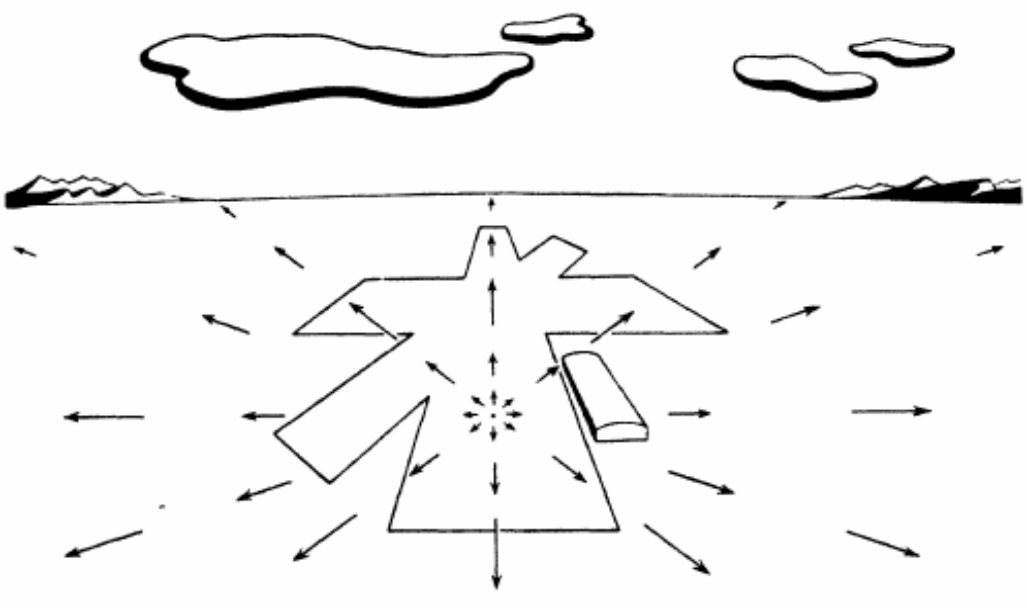


Figura 1. Flujo óptico al momento de un aterrizaje (Gibson, 1950)

La Figura 1 puede verse como una representación del flujo óptico generado cuando un piloto realiza un aterrizaje, donde cada flecha representa la velocidad y la dirección del movimiento que tiene cada parche de la imagen. En este ejemplo las flechas parten de un punto que es llamado ”Foco de expasión”, que es el punto hacia donde el observador se dirige y que aparentemente carece de movimiento, las flechas de mayor tamaño corresponden a puntos que se encuentran a

corta distancia ya que su movimiento es rápido, mientras que para las flechas de menor tamaño es todo lo contrario es decir, son puntos que se encuentran a mayor distancia con respecto al observador y que aparentemente su movimiento es lento.

2.2. Flujo Óptico

El flujo óptico es originado por el movimiento aparente de los objetos en una secuencia de imágenes. Se define $I(x, y, t)$ como una función de la intensidad de la imagen en un punto (x, y) en un tiempo t. Cuando este punto se mueve a lo largo de la secuencia, existe un vector **v** que describe su desplazamiento tanto horizontal como vertical. El objetivo es estimar un vector de velocidad (u, v) para cada punto en la imagen, tal que $I(x, y, t)$ y $I(x + u, y + v, t + 1)$ sean correspondientes.

Entre las aplicaciones que tiene, por mencionar algunas podemos encontrar seguimiento de objetos, detección de movimiento, segmentación de objetos en base a movimiento.

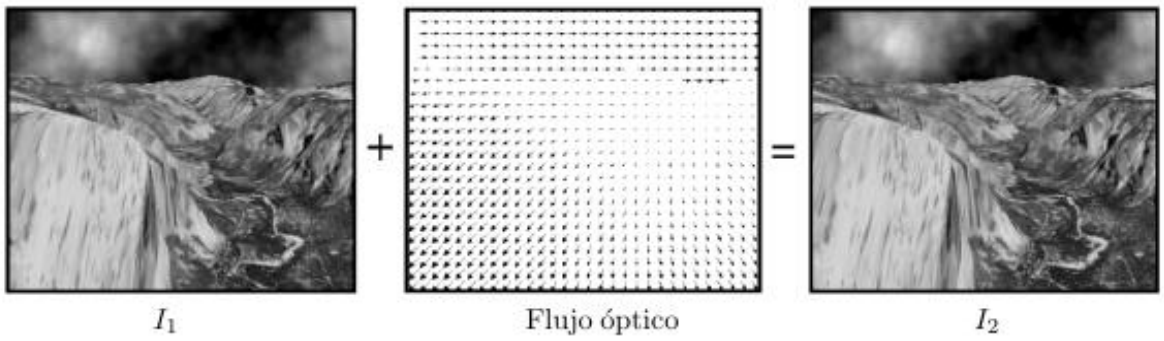


Figura 2. Flujo óptico. Secuencia Yosemite

Para la estimación del flujo óptico en la literatura se proponen algunas restricciones con el fin de modelar el comportamiento de dicho flujo y así obtener una estimación precisa. Una de ellas es la suposición de constancia de la intensidad de la imagen ó también conocida como restricción de conservación de los datos. Considerando una región en una secuencia de imágenes (Figura 3), la ubicación espacial de esta a lo largo de la secuencia puede cambiar debido a el movimiento de

los objetos dentro de la escena pero los valores de la intensidad deben permanecer constantes.

$$I(x+u, y+v, t+1) = I(x, y, t) \tag{1}$$

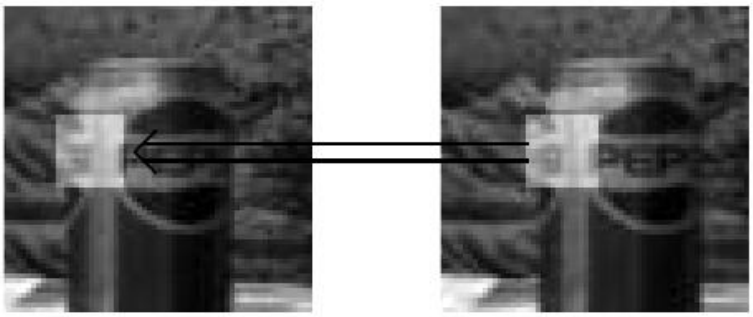


Figura 3. Ejemplo de la restricción de conservación de los Datos.

Esto es que la intensidad en la imagen en un punto (x, y) en un tiempo t, es la misma para la imagen en un instante $t+1$, en donde el punto (x, y) ha tenido un desplazamiento u en el eje horizontal y un desplazamiento v en el vertical. En la práctica esta restricción puede ser infringida al existir cambios en la iluminación de la escena o simplemente por ruido que pudiera generar el sensor.

Asumiendo que esta restricción se cumple para todos los pixeles en la imagen, se puede formular una función objetivo a minimizar de la siguiente forma:

$$E_D(\mathbf{u}, \mathbf{v}) = \sum_S (I(x+u, y+v, t+1) - I(x, y, t))^2 \tag{2}$$

en donde E_D es una suma que considera todos los píxeles de la imagen y es una función de los vectores de flujo **u** y **v** que corresponden a los desplazamientos horizontales y verticales.

Otra suposición que se hace con respecto al flujo dice que puntos vecinos en una región de la imagen pertenecen a un mismo objeto ó superficie y por lo tanto se espera que el flujo varie de manera suave.

$$U_p = U_n \quad n \in G(p) \tag{3}$$

Considerando la Figura 4, sea U un punto en la imágen, la Ecuación 3 nos dice que el movimiento que se tiene en U en la coordenada p es igual al que tiene U en una coordenada n, donde el punto en la coordenada n es un pixel que se encuentra dentro del vecindario del punto p.

Esta restricción en ocasiones también es violada debido a las discontinuidades que pueden presentarse en el movimiento de los objetos, como al existir solapamiento entre ellos y considerar que un conjunto de puntos pertenecen a un mismo objeto cuando en realidad podria pertenecer a otro y tener un movimiento completamente diferente.

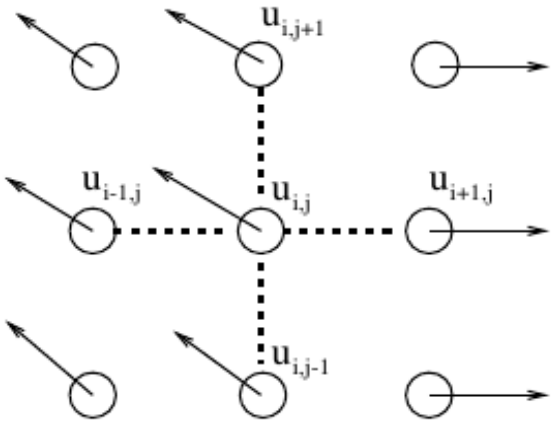


Figura 4. Discontinuidades en el movimiento dentro de un vecindario de píxeles (Black,1992)

De manera formal esta restricción puede ser escrita como:

$$E_S(\mathbf{u},\mathbf{v}) = \sum_{n\in G(s)} (u_s - u_n)^2 + \sum_{n\in G(s)} (v_s - v_n)^2 \tag{4}$$

Considerando ambas restricciones tenemos una función de minimizacion de energia:

$$E(\mathbf{u},\mathbf{v}) = E_D(\mathbf{u},\mathbf{v}) + \lambda E_S(\mathbf{u},\mathbf{v}) \tag{5}$$

De manera desarrollada

$$\begin{aligned} E(\mathbf{u},\mathbf{v}) = &\sum_{S} (I(x+u, y+v, t+1) - I(x,y,t))^2 \\ &+\lambda\left(\sum_{n\in G(s)} (u_s - u_n)^2 + \sum_{n\in G(s)} (v_s - v_n)^2 \right) \end{aligned} \tag{6}$$

en donde λ es un término de regularización que indica la influencia de la restricción de suavidad. La Ecuación (5) es la formulación clasica de la función objetivo de flujo óptico, para resolverla se

utiliza una aproximación por serie de Taylor.

$$E_D(\mathbf{u},\mathbf{v}) = \sum_S (I(x+u, y+v, t+1) - I(x,y,t))^2$$

$$dx = u, dy = v, dt = 1$$

$$I(x,y,t) + dx\frac{\partial}{\partial x}I(x,y,t) + dy\frac{\partial}{\partial y}I(x,y,t) + dt\frac{\partial}{\partial t}I(x,y,t) - I(x,y,t) = 0$$

Simplificando:

$$u\frac{\partial}{\partial x}I(x,y,t) + v\frac{\partial}{\partial y}I(x,y,t) + \frac{\partial}{\partial t}I(x,y,t) = 0$$

La ecuación anterior puede reescribirse como:

$$I_x u + I_y v + I_t = 0 \tag{7}$$

o bien

$$\nabla I \cdot \vec{v} = -I_t$$

en donde u, v son los vectores de desplazamiento horizontal y vertical en un punto en la imagen. I_x, I_y y I_t corresponde a las derivadas parciales de la intensidad de la imagen en x, y y t respectivamente. A esta ecuación se le conoce como la ecuación de restricción de flujo óptico.

Evaluando la Ecuación 7 en un solo punto en la imagen, tenemos una ecuación con dos incognitas (u, v), por lo que es necesario contar con más información. Esto da origen a lo que se conoce como ”Problema de apertura”, en el cual al observar el movimiento en un punto de un borde a través de una apertura, el movimiento que se tiene es ambiguo y solo se puede recuperar el movimiento que es perpendicular a la orientación de dicho borde.

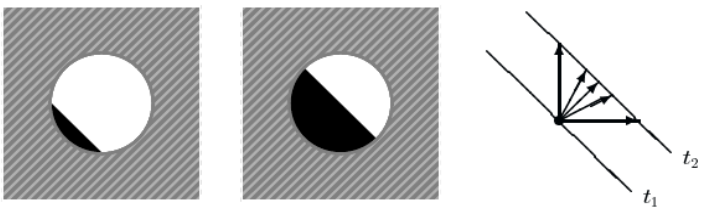


Figura 5. Problema de apertura (Mallot, 2000)

2.2.1. Clasificación de los métodos

Los métodos que existen dentro de la literatura para la estimación del flujo óptico pueden ser clasificados de acuerdo a (Barron, 1994) de la siguiente manera:

- Métodos Diferenciales.

  En ellos, se calcula el desplazamiento que tienen los píxeles usando derivadas espaciotemporales de las intensidades de la imagen. Uno de los métodos más representativos de esta clasificación es el propuesto por (Horn, 1981), el cual se abordará más adelante.

- Métodos basados en la correlación.

  En este tipo de métodos se realiza la busqueda de correspondencia utilizando pequeñas ventanas o patrones alrededor de cada píxel que componen la imagen, con la finalidad de maximizar alguna medida de similaridad. Algunos métodos que caben dentro de esta clasificación son los propuestos por Kalivas (1991), Kories (1986), Sutton (1983), Little (1988), por mencionar algunos.

- Métodos basados en la Frecuencia.

  Utilizan la transformada de Fourier para calcular el flujo óptico a través del dominio de la frecuencia. Algunos de los trabajos que destacan en esta clasificación son los de Watson (1985), Adelson (1985), Fleet (1990), Heeger (1988).

2.2.2. Método de Horn & Schunck

Uno los primeros trabajos en formalizar un método computacional para determinar el flujo óptico entre un par de imágenes fue el propuesto por (Horn, 1981). En el parten de la formulación clásica de la función objetivo del flujo óptico y proponen un método para optimizarla.

$$E(\mathbf{u},\mathbf{v}) = \sum_{S}(I_x u + I_y v + I_t)^2 + \lambda \sum_{n\in G(s)} \left((u_s - u_n)^2 + (v_s - v_n)^2 \right)$$

Para aproximar las derivadas parciales de la intensidad de la imagen (I_x, I_y, I_t), se utilizan diferencias finitas, donde el subíndice i, j corresponde a los ejes y, x en la imagen respectivamente y

k corresponde al tiempo.

$$
\begin{aligned}
I_x &\approx \frac{1}{4}\{I_{i,j+1,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i+1,j,k} + \\
&I_{i,j+1,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i+1,j,k+1}\} \\
I_y &\approx \frac{1}{4}\{I_{i+1,j,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i,j+1,k} + \\
&I_{i+1,j,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i,j+1,k+1}\} \\
I_t &\approx \frac{1}{4}\{I_{i,j,k+1} - I_{i,j,k} + I_{i+1,j,k+1} - I_{i+1,j,k} + \\
&I_{i,j+1,k+1} - I_{i,j+1,k} + I_{i+1,j+1,k+1} - I_{i+1,j+1,k}\}
\end{aligned}
\tag{8}
$$

Acorde al trabajo de Horn-Schunck, una manera de expresar la restricción de suavidad de la Ecuación (4), es minimizando el cuadrado de la magnitud del gradiente de los vectores de flujo.

$$
\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \qquad \text{y} \qquad \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{9}
$$

Ó también como la suma de los cuadrados del Laplaciano de las componentes en x y y del flujo, que son definidos como:

$$
\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \qquad \text{y} \qquad \nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \tag{10}
$$

Para realizar la aproximación del Laplaciano de u y v:

$$
\nabla^2 u \approx (\overline{u}_{i,j,k} - u_{i,j,k}) \quad \text{y} \quad \nabla^2 v \approx (\overline{v}_{i,j,k} - v_{i,j,k})
$$

Donde $\overline{u}$ y $\overline{v}$, son los promedios de u y v dentro de un vecindario y estan definidos como:

$$\begin{aligned}\overline{u}_{i,j,k} = \frac{1}{6}\{u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k}\} + \\ \frac{1}{12}\{u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}\}\end{aligned} \tag{11}$$

$$\begin{aligned}\overline{v}_{i,j,k} = \frac{1}{6}\{v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k}\} + \\ \frac{1}{12}\{v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}\}\end{aligned}$$

Para realizar la estimación del flujo óptico, Horn-Schunck propone un método iterativo por medio del método *Gauss-Seidel*.

$$u^{n+1} = \frac{\overline{u}^n - I_x[I_x\overline{u}^n + I_y\overline{v}^n + I_t]}{(\alpha^2 + I_x^2 + I_y^2)} \tag{12}$$

$$v^{n+1} = \frac{\overline{v}^n - I_x[I_x\overline{u}^n + I_y\overline{v}^n + I_t]}{(\alpha^2 + I_x^2 + I_y^2)}$$

donde el superíndice n denota en este caso el número de iteración, I_x, I_y y I_t son las derivadas parciales de la intensidad de la imagen en x, y y t respectivamente; $\overline{u}$ y $\overline{v}$ son promedios que son calculados acorde a la Ecuación (10) y α es una constante que indica la influencia de la restricción de suavidad.

Algoritmo 1 Método de Horn-Schunck

Descripción: Realiza la estimación del flujo óptico entre dos imágenes mediante el método de Horn-Schunck.

Entrada:

- $Im1, Im2$: Dos imágenes subsecuentes en escala de grises.
- α : Parámetro que indica la influencia de la restricción de suavidad.
- $nIter$: Número de iteraciónes.

Variables:

- u, v : Componentes de flujo óptico en dirección x, y respectivamente.
- I_x, I_y, I_t : Derivadas parciales de la intensidad de la imagen con respecto a x, y, t.
- $\overline{u}$ y $\overline{v}$: Promedios de u y v dentro de un vecindario.
- n : Contador de iteraciónes.

Salida: (u, v) Componentes del flujo óptico para la secuencia de entrada.

Inicio

$u \leftarrow 0$
$v \leftarrow 0$
$n \leftarrow 0$
$I_x, I_y, I_t \leftarrow$ calcular mediante (8) $\rightarrow Im1, Im2$
while $n < nIter$ **do**
    $\overline{u}, \overline{v} \leftarrow$ Calcula $\overline{u}, \overline{v}$ mediante (11)
    $u, v \leftarrow$ Calcula u, v mediante (12)
    $n \leftarrow n + 1$
end while
return $[u, v]$

2.2.3. Método de Lucas & Kanade

Otro de los métodos clásicos para realizar la estimación del flujo óptico es el que fue propuesto por (Lucas, 1981). Este parte de que la Ecuación (7) se cumple para todos los pixeles dentro de un vecindario de tamaño n que tiene como centro un pixel p, de tal forma que se resuelve el sistema de ecuaciones resultantes para dicho vecindario y el vector resultante **v** se asigna a el punto p.

Para un vecindario de tamaño n se tendría un sistema de ecuaciones de la siguiente manera:

$$\begin{aligned} I_{x1}u &+ I_{y1}v = -I_{t1} \\ I_{x2}u &+ I_{y2}v = -I_{t2} \\ &\vdots \\ I_{xn}u &+ I_{yn}v = -I_{tn} \end{aligned}$$

en donde I_{xi}, I_{yi} y I_{ti}, corresponden a las derivadas parciales de la intensidad con respecto a x, y, t respectivamente evaluadas en el punto i de la imagen. Como podemos observar se tiene un sistema de ecuaciones sobredeterminado, ya que el número de ecuaciones sobrepasa al número de incógnitas. Reescribiendo el sistema de ecuaciones de manera matricial de la forma $A\mathbf{v} = \mathbf{b}$:

$$A = \begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{xn} & I_{yn} \end{bmatrix}, \qquad \mathbf{v} = \begin{bmatrix} u \\ v \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{tn} \end{bmatrix}$$

El vector **v** se obtiene de la siguiente manera:

$$\begin{aligned} A\mathbf{v} &= \mathbf{b} \\ A^T A\mathbf{v} &= A^T\mathbf{b} \\ \mathbf{v} &= (A^T A)^{-1} A^T\mathbf{b} \end{aligned}$$

Donde:

$$A^T A = \begin{bmatrix} \sum_{i=1}^{n} I_{xi}^2 & \sum_{i=1}^{n} I_{xi} I_{yi} \\ \sum_{i=1}^{n} I_{xi} I_{yi} & \sum_{i=1}^{n} I_{yi}^2 \end{bmatrix}^{-1} \quad A^T\mathbf{b} = \begin{bmatrix} \sum_{i=1}^{n} I_{xi} I_{ti} \\ \sum_{i=1}^{n} I_{yi} I_{ti} \end{bmatrix}$$

Para dar solución a la ecuación $\mathbf{v} = (A^T A)^{-1} A^T\mathbf{b}$, la matriz $A^T A$ debe de ser invertible. En la práctica, suele usarse la Pseudoinversa de Moore-Penrose para realizar esta estimación cuando el sistema de ecuaciones lineales no posee solución única.

Algoritmo 2 Método de Lucas-Kanade

Descripción: Realiza la estimación del flujo óptico entre dos imágenes mediante el método de Lucas-Kanade.

Entrada:

- Im1, Im2 : Dos imágenes subsecuentes en escala de grises.
- w : Número entero que indica el tamaño del vecindario.

Variables:

- u, v : Arreglos bidimensionales que contendrán las componentes de flujo óptico en dirección x, y respectivamente.
- I_x, I_y, I_t : Derivadas parciales de la intensidad de la imagen con respecto a x, y, t.
- I_{xw}, I_{yw}, I_{tw} : Arreglos para las derivadas parciales de la intensidad de la imagen con respecto a x, y, t dentro de un vecindario.
- i, j : Contadores.
- A, b, uv : Arreglos.

Salida: (u, v) Componentes del flujo óptico para la secuencia de entrada.

Inicio

$u, v \leftarrow Im1.size$
$w \leftarrow round(w/2)$
$i, j \leftarrow 0$
$I_x, I_y, I_t \leftarrow$ calcular mediante (8) $\rightarrow Im1, Im2$
for $i = w + 1$ **to** $Im1.rows - w$ **step** 1 **do**
    for $j = w + 1$ **to** $Im1.cols - w$ **step** 1 **do**
        $I_{xw} \leftarrow I_x(i - w : i + w, j - w : j + w)$
        $I_{yw} \leftarrow I_y(i - w : i + w, j - w : j + w)$
        $I_{tw} \leftarrow I_t(i - w : i + w, j - w : j + w)$
        $b \leftarrow -I_{tw}$
        $A \leftarrow [I_{xw} I_{yw}]$
        $uv \leftarrow pseudoInv(A) * b$
        $u(i, j) \leftarrow uv[1]$
        $v(i, j) \leftarrow uv[2]$
    end for
end for

2.2.4. Flujo óptico usando un esquema multi-resolución.

Una desventaja de los métodos para calcular el flujo óptico como el de Horn-Schunck y Lucas-Kanade, es que en su formulación supone que los desplazamientos que se llevan acabo dentro de la escena son pequeños, esta suposición dependiendo del tipo de movimiento puede o no cumplirse, lo que deriva en obtener una estimación errónea del flujo.

Para tratar con este problema se han implementado algunas estrategias como lo es la estimación de flujo óptico sobre un esquema piramidal. En este esquema, se contruye una piramide en donde la base corresponde a las imágenes de la secuencia en su dimensión original. Los niveles superiores se conforman de una versión dimensionalmente reducida dado cierto factor de re-escalamiento.

La estimación del flujo óptico se comienza en el nivel superior y consiste en un proceso iterativo proyectando dicha estimación al siguiente nivel realizando el correspondiente cambio de resolución hasta llegar a la base de la piramide, en donde se obtendría una estimación del flujo con las dimensiones originales para la secuencia de entrada.

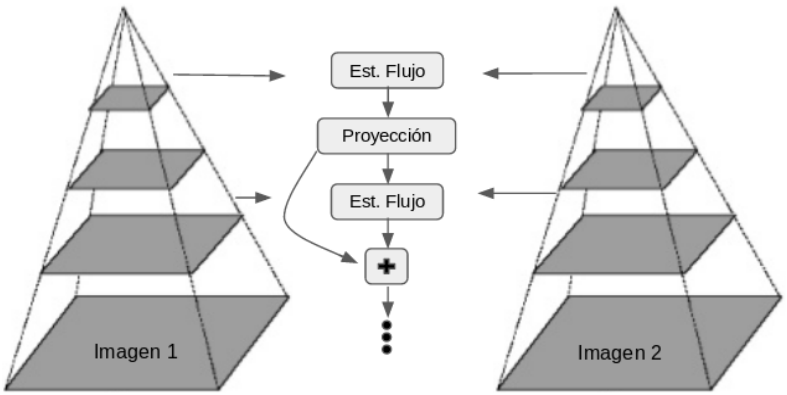


Figura 6. Esquema Multi-resolución.

Este esquema puede ser aplicado a métodos para estimar el flujo óptico como el del Horn-Schunck, con la finalidad de obtener una estimación precisa.

Algoritmo 3 Obtener número de niveles

Descripción: Devuelve el número de niveles que tendrá la pirámide considerando las dimensiones de la secuencia de entrada, asegurandosé que la dimensión final sea divisible entre 2^n

Entrada:

- Im : Marco perteneciente a la secuencia.

Variables:

- $nLvl$: Número de niveles para la pirámide.
- i, j, n : Contadores.

Salida: (u, v) Número de niveles para la pirámide.

Inicio

$i, j \leftarrow 1$
$n \leftarrow 0$
while 1 **do**
    $n \leftarrow 2^i$
    if $Im.cols \,\%n == 0$ **then**
        $i \leftarrow i + 1$
    else
        $break$
    end if
end while
$n \leftarrow 0$
while 1 **do**
    $n \leftarrow 2^j$
    if $Im.rows \,\%n == 0$ **then**
        $j \leftarrow j + 1$
    else
        $break$
    end if
end while
if $i < j$ **then**
    $nLvl \leftarrow i$
else
    $nLvl \leftarrow j$
end if
return $nLvl$

Algoritmo 4 Genera pirámide

Descripción: Genera esquema piramidal con la secuencia de entrada
Entrada:

- $Im1, Im2$: Dos imágenes subsecuentes en escala de grises.
- $nLvl$: Número de niveles para la piramide.

Variables:

- $Im1p, Im2p$: Dos arreglos, que representan cada marco de la secuencia de entrada en el esquema piramidal.
- i, n : Contadores.

Salida: $Im1p, Im2p$
Inicio

$n \leftarrow 1$
while $n < nLvl$ **do**
    $Imp1[n] \leftarrow resize(Im1) * 0.5$
    $Imp2[n] \leftarrow resize(Im2) * 0.5$
    $n \leftarrow n + 1$
end while
$ImgP[1] \leftarrow Im1p$
$ImgP[2] \leftarrow Im2p$
return $ImgP$

Algoritmo 5 Método de Horn-Schunck con esquema piramidal

Descripción: Realiza la estimación del Flujo Óptico entre dos imágenes mediante el método de Horn-Schunck usando un esquema multiresolución.
Entrada:

- $Im1, Im2$: Dos imágenes subsecuentes en escala de grises.
- a : Parámetro que indica la influencia de la restricción de suavidad.
- $nIter$: Número de iteraciones.

Variables:

- u, v : Componentes de flujo óptico en dirección x, y respectivamente.
- $uPrev, vPrev$: Componentes de flujo óptico en dirección x, y respectivamente obtenidos en el nivel previo en la pirámide.
- $nLvl$: Número de niveles para la pirámide.
- $ImgP$: Arreglo que contiene la secuencia en el esquema piramidal.
- i : Contador.

Salida: (u, v) Componentes del flujo óptico para la secuencia de entrada.
Inicio

$nLvl \leftarrow$ **Algoritmo 3**$(Im1)$
$ImgP \leftarrow$ **Algoritmo 4**$(Im1, Img2, nLvl)$
$[u, v] \leftarrow$ **Algoritmo1**$(ImgP[1, nLvl], ImgP[2, nLvl], nIter, a)$
$i \leftarrow nLvl - 1$
while $i > 0$ **do**
    $u \leftarrow resize(u) * 2$
    $v \leftarrow resize(v) * 2$
    $[uPrev, vPrev] \leftarrow$ **Algoritmo1**$(ImgP[1, i], ImgP[2, i], nIter, a)$
    $[u, v] = [u, v] + [uPrev, vPrev]$
    $i \leftarrow i - 1$
end while
return $[u, v]$

2.2.5. Representación visual del flujo óptico

Existen dos tipos de representación visual para el flujo óptico estimado en una secuencia de imágenes. Una de estas, es por medio de flechas representando los vectores de flujo estimados para cada uno de los pixeles de la imagen. En la Figura 7 puede observarse este tipo de visualización para el flujo óptico estimado en la secuencia Dimetrodon de la base de datos de Middlebury.

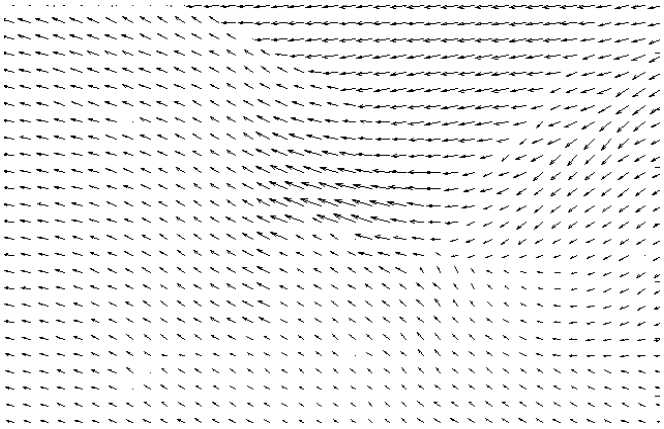


Figura 7. Representación del flujo por medio de vectores

Otro tipo de visualización es por medio de un código de colores, en donde la tonalidad indica la dirección y la saturación la magnitud del vector de flujo en ese punto de la imagen. En la Figura 8 puede observarse este tipo de visualización para el flujo óptico estimado en la secuencia Dimetrodon.

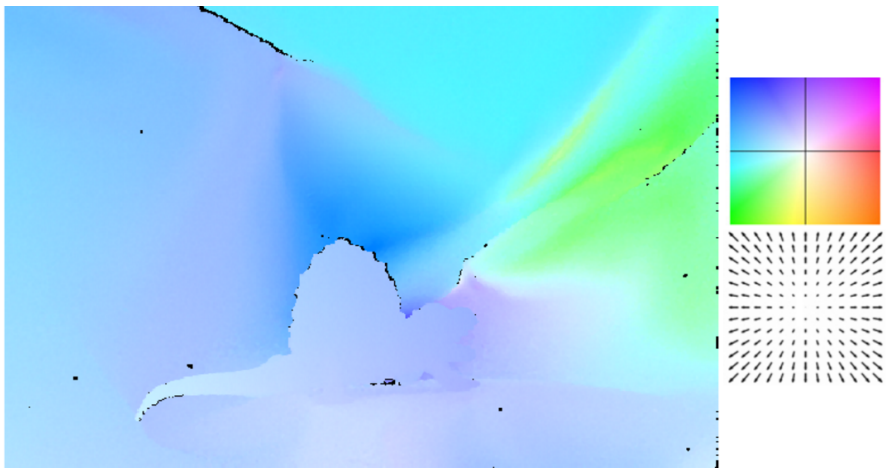


Figura 8. Representación del flujo por medio de código de colores

2.3. Cómputo evolutivo y flujo óptico

En la literatura se han abordado problemas de visión por computadora utilizando técnicas de cómputo evolutivo para resolverlos. Métodos para la estimación de movimiento se pueden encontrar en trabajos propuestos por Li (1999), Gong (2002) y Zavala-Romero (2011) por mencionar algunos.

Un ejemplo de cómputo evolutivo y flujo óptico es el método propuesto por (Tagliasacchi, 2006). Éste método consta principalmente de dos etapas que se describen a continuación:

- En la primera etapa se realiza una segmentación de la imágen agrupando aquellos píxeles cuyos vecinos compartan valores similares de intesidad, ya que se asume que el flujo varia de manera suave y que píxeles correspondientes a una misma región espacial poseén un desplazamiento similar.
- Después mediante un algoritmo genético, se buscan seis parámetros para un modelo de movimiento afín para cada uno de los puntos dentro de las regiones obtenidas en la fase de segmentación.

Una vez que los parámetros son conocidos, las componentes del flujo para cualquier punto *x,y* estan dados por:

$$v_x(x_i, y_i) = a_1 + a_3\frac{x_i}{C_x} + a_5\frac{y_i}{C_y}$$
$$v_y(x_i, y_i) = a_2 + a_4\frac{x_i}{C_x} + a_6\frac{y_i}{C_y}$$

En donde $v_x(x_i, y_i)$ y $v_y(x_i, y_i)$ corresponden a las componentes del flujo en la dirección de x, y respectivamente. El vector **a** = $(a_1, a_2, a_3, a_4, a_5, a_6)$ contiene los parámetros del modelo de movimiento afín y C_x y C_y son las coordenadas de los centroides de cada una de las regiones segmentadas.

La función objetivo consiste en encontrar el vector **a** que minimize el promedio de la diferencia entre los niveles de intensidad del marco actual y el marco de referencia siguiendo la trayectoria descrita por el flujo óptico calculado.

$$\mathbf{a} = min\frac{1}{M}\Sigma_{i=1}^{M}|I(x_i, y_i, t) - I(x_i - v_x(x_i, y_i), y_i - v_y(x_i, y_i), t-1)|^2$$

En donde $I(x, y, t)$ es la función de intensidad de la imágen en un punto x, y en el tiempo t y M es el total de píxeles correspondientes a una región

Éste método reporta un error ángular promedio de 12.13° en la estimación del flujo óptico para la secuencia Yosemite, que corresponde al conjunto de prueba de la base de datos de Middlebury.

Los *Algoritmos Evolutivos* son útiles cuando se requiere resolver un problema cuyo espacio de soluciones sea muy grande, si este se delimita de manera razonable es posible encontrar soluciones que con los métodos tradicionales normalmente no se encontrarian. En el capítulo siguiente se tratan conceptos de la *Programación Genética* (Koza, 1992), sobre la cual se basa la estrategia empleada en este trabajo.

2.4. Medidas de evaluación en la estimación de flujo óptico.

Para el proceso de evaluación de resultados en este trabajo se utilizará el conjunto de secuencia de imágenes de Middlebury para flujo óptico. Esta base de datos consta de un conjunto de entrenamiento y otro de prueba cada uno de doce secuencias disponibles tanto en escala de grises como a color, en donde para ocho de ellas del conjunto de entrenamiento se encuentra disponible los valores de referencia del flujo óptico (*ground-truth*).

En esta base de datos se reportan principalmente dos medidas de precisión, originalmente propuestas por (Barron, 1994) que son:

- Error de punto final (End-Point Error), que es una distancia Euclidiana:

$$EPE = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(u_i - u_i^{gt})^2 + (v_i - v_i^{gt})^2} \tag{13}$$

- Error angular promedio (Average Angular error), que es el ángulo formado entre los vectores estimados y los vectores de referencia:

$$AAE = \frac{1}{N}\sum_{i=1}^{N}\arccos\left(\frac{u_i u_i^{gt} + v_i v_i^{gt}}{\sqrt{u_i^2 + v_i^2 + 1}\sqrt{u_i^{2,gt} + v_i^{2,gt} + 1}}\right) \tag{14}$$

donde N es el total de píxeles en la imagen, u, v son los vectores de desplazamiento para el píxel *i* del flujo estimado y u^{gt}, v^{gt} el de referencia (*ground-truth*).

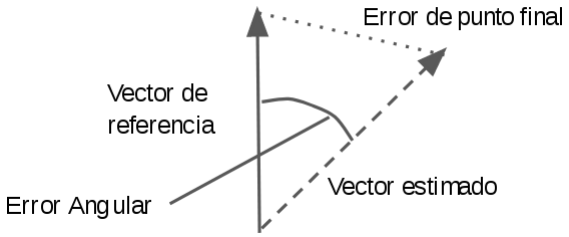


Figura 9. Representación gráfica de los errores AAE y EPE

Capítulo 3. Programación Genética

3.1. Cómputo Evolutivo

El Cómputo Evolutivo (*EC, del inglés Evolutionary Computing*) es un área de la inteligencia artificial, la cual está inspirada en la teoría de la evolución presentada por Charles Robert Darwin en ”El origen de las especies” (Darwin, 1859), que explica el proceso de adaptación de las especies por medio de la selección natural, donde se favorecen a aquellos individuos que se adaptan con mayor facilidad a su entorno y por consecuencia serán los que sobrevivirán.

Esta técnica bio-inspirada provee de una metodología para resolver problemas de optimización de manera automática, siguiendo una estrategia de busqueda guiada utilizando un proceso de prueba y error que se basa en el paradigma de la evolución artificial.

Dentro de la literatura se destacan cuatro subramas en computo evolutivo que son:

- Algoritmos Genéticos, propuesto por (Holland, 1975).
- Estrategías Evolutivas (Rechenberg, 1973), (Schwefel, 1995).
- Programación Evolutiva, (Fogel, 1966).
- Programación Genética, propuesto por (Koza, 1992).

En el presente trabajo se utilizó la metodología presentada por Koza (1992), que se conoce como programación genética, la cual se aborda con más detalle en las siguientes secciones.

3.2. Conceptos básicos de la programacion genética

La programación genética o GP (*del inglés Genetic Programming*), fue desarrollada por Koza (1992), la cual incluye un conjunto de técnicas de computo evolutivo que permite resolver problemas de manera automatica.

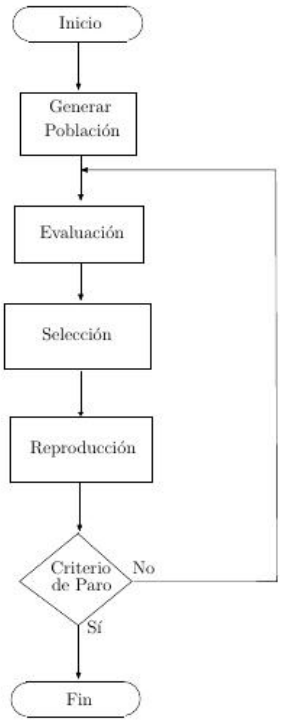


Figura 10. Diagrama de flujo de un algoritmo evolutivo

En cuanto a la estructura de un programa de GP, podemos decir que es muy similar al resto de técnicas de EC pero se distingue por la forma de representar las soluciones, ya que en un programa de GP los individuos es decir, las posibles soluciones al problema son codificados usando una estructura de árbol. Cada individuo se contruye en base a dos conjuntos finitos, los nodos internos pertenecen a un conjunto denominado *Funciones* y para los nodos que se encuentran en los últimos niveles de los árboles tambien conocidos como hojas se toman de otro conjunto llamado *Terminales*.

La definición de estos conjuntos es de suma importancia, ya que estos delimitan el espacio de búsqueda de soluciones que tendrá el programa de GP. A su vez podemos identificar más otros dos aspectos importantes que son: la definición de una función objetivo con la cual se evaluará la aptitud de cada individuo (posible solución al problema) asi como un criterio de terminación para el algoritmo, el cual comúnmente puede ser alcanzar un cierto límite de generaciones o al satisfacer una condición previamente definida.

3.2.1. Funciones y terminales

La definición de estos conjuntos depende directamente del tipo de problema que se este abordando usando GP. Algunos de los elementos por los cuales puede estar formado el conjunto de funciones son:

- Operadores Aritméticos: {+,-,*,/,etc.}
- Operadores Booleanos: {AND, OR, NOT}
- Funciones Matemáticas: {Sin, Cos, Tan, etc}
- Sentencias Condicionales: {If, Then, Else, Case}
- Sentencias de Ciclos: {While...Do,Repeat...Until, For,etc }
- Sentencias de Control de Transferencia: {Go to, Call, Jump}

Los elementos pertenecientes tanto al conjunto de Funciones como al de terminales, son denominados *nodos* cuando la representación de los programas es en forma de árboles. El conjunto de terminales esta formado por:

- Variables, las cuales son las entradas al programa.
- Constantes, que como su nombre lo dice son elementos que no cambian su valor a lo largo de la ejecución del programa.

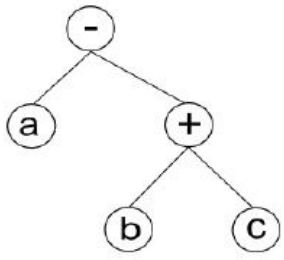


Figura 11. Rama de un árbol

En este caso los nodos {a,b,c} son parte del conjunto de terminales, mientras que {-,+} son parte del conjunto de Funciones.

3.2.2. Métodos de inicialización de la población

Basandonos en la Figura 11, el primer paso que se debe llevar acabo en un programa de GP, es la inicialización de la población, es decir generar un conjunto de tamaño P, con estructuras en forma de árboles los cuales tendrán una *máxima profundidad* previamente definida en el programa.

Para inicializar la estructura de los árboles existen algunos métodos los cuales se explican a continuación.

3.2.2.1. Método Completo y Crecimiento

En el método Completo *(del inglés full)*, cada rama que compone el arbol tiene una profundidad máxima que podemos denotar como D_{max} y se componen de la siguiente manera:

- Los nodos que se encuentran a un nivel $d < D_{max}$, son elegidos aletoriamente del conjunto de Funciones F.
- Los nodos del nivel $d = D_{max}$, aleatoriamente son elegidos del conjunto de Terminales T.

Las ramas de los árboles generados mediante el método de Crecimiento *(del inglés grow)* a diferencia del método Completo, pueden tener una profundidad diferente siempre y cuando sean $\leq D_{max}$.

- Los nodos que se encuentran a un nivel $d < D_{max}$, se eligen aleatoriamente del conjunto $F \cup T$.
- Similar a el método Completo, los nodos que se encuentran a un nivel $d < D_{max}$, son elegidos aletoriamente del conjunto de Funciones F.

3.2.2.2. Método Expansión Mitad y Mitad

En el método Expansión Mitad y Mitad (*del inglés "ramped half-and-half"*), los árboles son inicializados usando tanto el método *Completo* y *Crecimiento*, eligiendo cada método en base a una probabilidad. Cada individuo sigue generándose a partir de los conjuntos *T* y *F* y una máxima produndidad D_{max}. Ésto se hace con la finalidad de que exista una mayor diversidad entre los individuos que componen la población.

3.3. Aptitud y selección

Siguiendo con el diagrama de la Figura 10, el siguiente paso es la evaluación de la aptitud (*Fitness en inglés*) de los individuos que componen la población en base a una *función objetivo*. La aptitud de cada uno de los individuos se espera que mejore conforme avance el proceso de evolución artificial, y es con ésta que se realiza la selección y se determinará cuales individuos entrarán al proceso de combinación, cuáles sobrevivirán y cuáles no.

3.3.1. Operadores genéticos

Una vez obtenida la población inicial así como la aptitud de cada uno de los individuos, se continua el proceso evolutivo aplicando sobre dicha población los operadores genéticos de *cruza* y *mutación*, los cuales se explican a continuación.

3.3.1.1. Cruza

También llamado recombinación, elige dos individuos de la población que son nombrados *padres* y a partir de estos se realizará un intercambio de información lo que genera otros nuevos individuos que son denominados *hijos*. El objetivo de este proceso es que si los padres seleccionados poseen una buena aptitud en base a la función objetivo, se espera que las nuevas soluciones obtenidas (hijos) tengan un mejor rendimiento. Así conforme el proceso de evolución artificial avance se espera encontrar individuos los cuales brinden cada vez mejores soluciones al problema que se está abordando.

En un programa de GP, el operador de cruce más común se realiza de la siguiente forma:

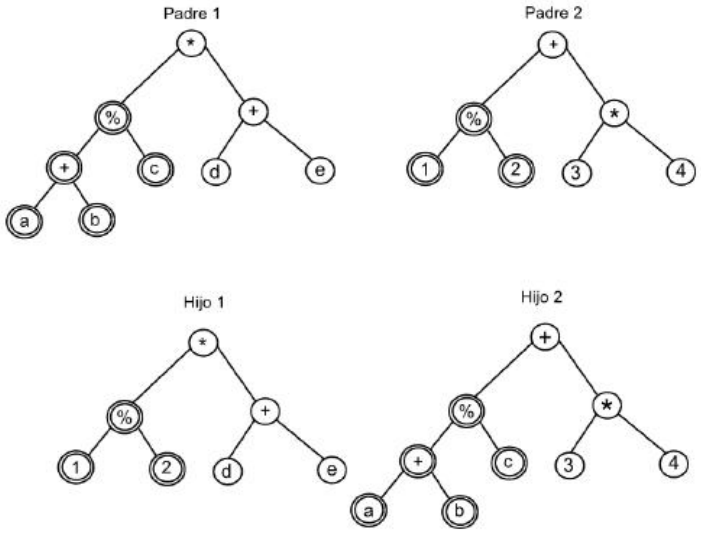


Figura 12. Ejemplo del operador cruce

- De la población, se seleccionan dos individuos lo cual puede ser de manera aleatoria o usando algún otro método de selección.
- En cada uno de los padres, se selecciona aleatoriamente un nodo el cual será el punto de cruce. Cada nodo posee la misma probabilidad de ser seleccionado.
- Por último, se intercambian los subárboles que se desprenden de los nodos seleccionado con anterioridad

Dando origen así a dos nuevos hijos. Cabe mencionar que la profundidad de los árboles puede superar a la de los padres después de este paso.

3.3.1.2. Mutación

Este operador es aplicado solo a un individuo de la población que es seleccionado mediante un proceso aleatorio. Por lo regular la probabilidad de que se aplique este operador es demasiado pequeña, de esta forma se busca incrementar la diversidad de los individuos dentro de la población

introduciendo nueva información y asi sirviendo de mecanismo que ayude al proceso evolutivo a poder escapar de posibles óptimos locales.

La mutación más común que se realiza a un inviduo es remplazando un subárbol elegido aleatoriamente por otro que es generado de igual manera.

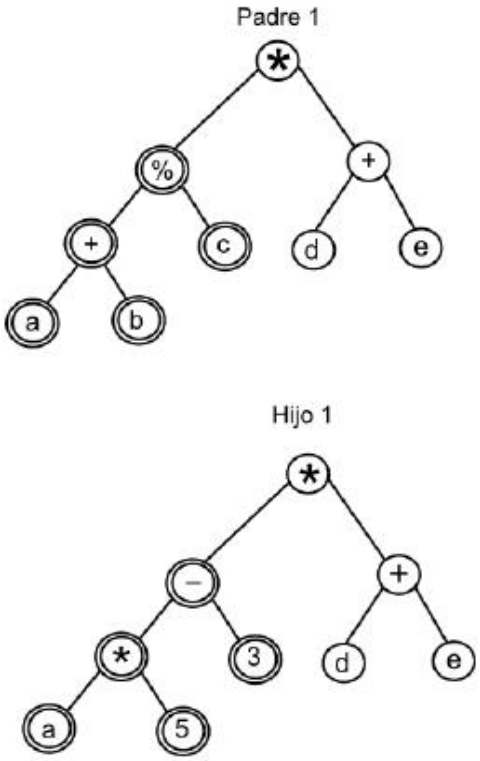


Figura 13. Ejemplo del operador mutación

Capítulo 4. Desarrollo de la propuesta

En este capítulo, se describe el algoritmo propuesto para el diseño automático de operadores que actuarán sobre la estimación de flujo óptico con la finalidad de obtener una precisión mayor a la de los métodos existentes en la literatura propuestos por expertos. Primeramente se describen algunos conceptos de funciones de transformación espaciales en imágenes que fueron empleadas, después se abordan otros aspectos del algoritmo como lo es el espacio de búsqueda que es formado por el conjunto de funciones y terminales propuesto; la forma de evaluación de las soluciones candidatas encontradas por el algoritmo y finalmente se describen detalles correspondientes a la implementación.

4.1. Función de transformación de imagen

Dentro de las funciones de transformación de imágenes podemos encontrar las funciones de transformación espaciales, este tipo de transformaciones tienen el propósito de modificar las propiedades geométricas de la imagen e idealmente, los valores de la intensidad entre la imagen sobre la que actúa la transformación y la imagen resultante deben de permanecer iguales.

Una imagen digital **I(x)** puede definirse como:

$$\mathbf{I(x)} : \mathcal{D} \subseteq \mathbb{R}^n \rightarrow \mathcal{C} \subseteq \mathbb{R}^m$$
$$\mathbf{x} \mapsto \mathbf{I(x)}$$

en donde m indica el número de canales o bandas que conforman la imagen (por ejemplo, 3 en el caso de una imagen RGB, 1 para una imagen en escala de grises, etc.) y n indica la dimensión de ésta.

Para este trabajo nos centraremos en un tipo de transformaciónes que son conocidas como *Funciones de mapeo*. Este tipo de funciones requieren como entrada una imagen y dado un vector de parámetros θ, toma puntos que la componen y los proyecta a otro plano. Para una imagen en 2D se puede definir como :

$$\mathbf{T}_\theta\mathbf{(x)} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$$
$$\mathbf{(x;}\theta\mathbf{)} \mapsto \mathbf{T}_\theta\mathbf{(x)}$$

en donde θ es un vector que indica los parámetros de la transformación y **x** es el punto a ser mapeado. Podemos mencionar dos tipos de mapeo: mapeo en dirección hacia adelante, y mapeo en dirección hacia atrás

4.1.1. Mapeo hacia adelante

El funcionamiento de este tipo de mapeo resulta ser sencillo, para cada punto en la imagen de entrada se toma el valor de su intensidad y éste es ubicado en la posición *(x,y)* que indique la función de mapeo en el plano de salida.

A continuación se presenta el algoritmo que describe este proceso.

Algoritmo 6 Mapeo hacia adelante — forwardMapping()

Descripción: Algoritmo de la función de mapeo hacia adelante.
Entrada:

- img : Arreglo que representa una imagen.
- $T()$: Función que mapea los puntos de un espacio de entrada a otro.

Variables:

- $[xy]$: Arreglo que contiene las coordenadas de un punto.

Salida: $imgT$: Arreglo que representa la imagen transformada.
Inicio

```
1: for i ← 1 to img.rows step 1 do
2:     for i ← 1 to img.cols step 1 do
3:         [xy] ← T(img(i, j))
4:         imgT(xy[1], xy[2]) ← img(i, j)
5:     end for
6: end for
7: return imgT
```

Este método de mapeo presenta algunas desventajas, y es que dependiendo de la función de mapeo $\mathbf{T}_\theta$ puede existir el caso en el que más de un punto proveniente de la imagen de entrada sea asignado a una misma posición en el plano de salida (*"solapamientos"*). Tambien puede presentarse el caso en el que la imagen de salida muestre algunos *"huecos"* por no haberle asignado a una posición un valor de intensidad.

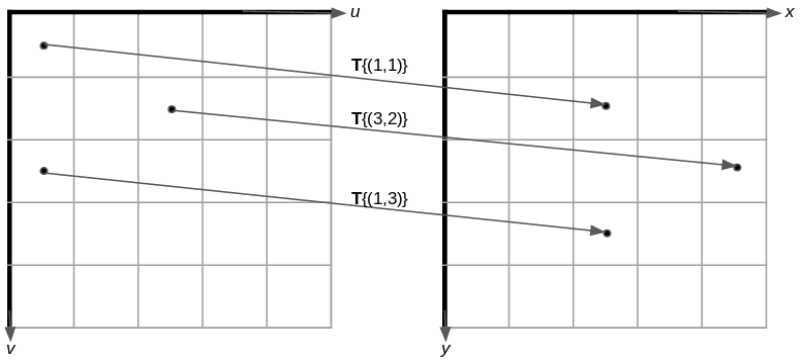


Figura 14. Mapeo hacia adelante

Con la finalidad anular las desventajas que presenta el mapeo hacia adelante, existe otro tipo de estrategia que es el *mapeo hacia atrás* o también conocido como *mapeo inverso*.

4.1.2. Mapeo hacia atrás

Este método de mapeo, para cada punto en el plano de salida (x, y) aplica la función de tranformación inversa $\mathbf{T}^{-1}$ para conocer la posición correspondiente de este punto en el plano de entrada con coordenada (u, v)

$$\mathbf{I}_{src}(u_i, v_i) = \mathbf{T}^{-1}(\mathbf{I}_{dst}(x_i, y_i))$$

Una vez conocida esta posición, se consideran los puntos cercanos a esta y usando un método de interpolación se apróxima el valor de intensidad. Finalmente este valor de intensidad es asignado al punto $\mathbf{I}_{dst}(x_i, y_i)$ en el plano de salida.

Con este método se pretenden evitar los solapamientos y los huecos que pudiesen presentarse usando un método como el mapeo hacia adelante; aunque es necesario elegir un método de interpolación asi como de definir alguna regla para cuando exista el caso en el que el punto referenciado por $\mathbf{I}_{dst}(x_i, y_i)$, no se encuentre dentro del plano de entrada.

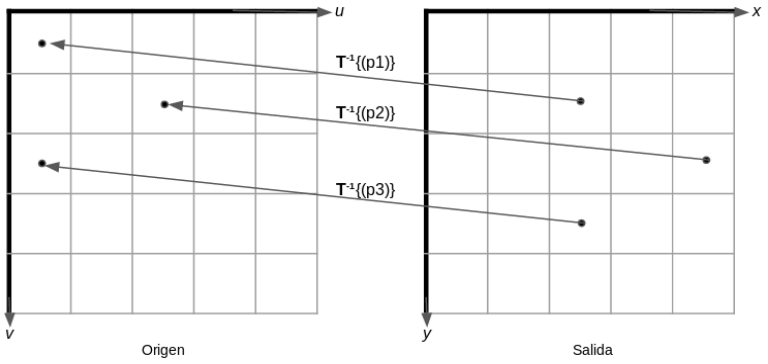


Figura 15. Mapeo hacia atrás

Algoritmo 7 Mapeo hacia atrás — inverseMapping()

Descripción: Algoritmo de la función de mapeo hacia atrás.
Entrada:

- img : Arreglo que representa una imagen.
- $T()$: Función que mapea los puntos de un espacio de entrada a otro.

Variables:

- $[xy]$: Arreglo que contiene las coordenadas de un punto.

Salida: $imgT$: Arreglo que representa la imagen transformada.
Inicio

1: **for each** *p* **in** *imgT* **do**
2: $[xy] \leftarrow T^{-1}(p)$
3: $imgT(p) \leftarrow img(xy[1], xy[2])$
4: **end for each**
5: **return** $imgT$

4.2. Descripción del algoritmo propuesto

Los métodos abordados anteriormente para la estimación de flujo óptico que son los propuestos por Horn-Schunck y Lucas-Kanade hacen uso de las secuencias de imágenes en escala de grises. Otros métodos existentes en la literatura no se limitan a esto como lo es el propuesto por (Aires, 2008), en el hacen uso de imágenes multicanal en distintos formatos por ejemplo RGB, HSV, YUV; lo que sugiere que puede emplearse adicionalmente la información de color para intentar obtener una mejor estimación del flujo. Considerando ésto, en el presente trabajo se realizarón dos versiones del algoritmo propuesto, el primero de estos consiste en generar de manera automática operadores para la corrección del flujo utilizando únicamente secuencias de

imágenes en escala de grises. La segunda versión integra la información que proveen diferentes bandas de color correspondientes a las secuencias de imágenes para la estimación del flujo y de igual manera genera de manera automática operadores para la corrección del flujo con la finalidad de obtener una estimación de mayor precisión.

4.2.1. Primera propuesta

El diseño de la primera versión del algoritmo propuesto consta a grandes rasgos de dos pasos: una estimación previa del flujo óptico y posteriormente la generación automática de operadores para la corrección del flujo. Para la primera parte se realiza una estimación previa de flujo óptico utilizando la secuencia en escala de grises, para ésto se emplea el método propuesto por Lukas-Kanade en su enfoque piramidal; se eligió este método debido a la simplicidad de implementación. Por otro lado, la generación automática de los operadores se realiza a través del proceso evolutivo, los cuales aplicados a la estimación previamente obtenida se busca que brinden un mejor desempeño en cuanto a la precisión en la estimación de los vectores de flujo.

De acuerdo a la Figura 10, el primer paso en un algoritmo evolutivo es la generación de la población. En este trabajo se utilizó una estrategia conocida como *programación cerebral* (Dozal, 2014). En esta técnica los individuos o posibles soluciones son definidos por arreglos con diversos árboles y no con uno solo, a diferencia de los algoritmos de GP. Dichos individuos son formados por conjuntos de operaciones básicas definidos en el capítulo 3 como conjunto de *funciones y terminales*.

Para este caso en particular cada posible solución puede tener como máximo una cantidad de seis árboles, donde cada árbol representa un operador el cual actuará sobre la estimación previa del flujo óptico con la finalidad de mejorar su precisión. Este número es generado de manera aleatoria para cada uno de los individuos de la población durante el proceso de inicialización. En el Algoritmo 8 se describe de manera general esta primera versión del algoritmo propuesto.

Algoritmo 8 Algoritmo de GP - proceso evolutivo

Descripción: Algoritmo que describe el proceso evolutivo.
Entrada:

- $Im1, Im2$: Dos imágenes subsecuentes en escala de grises.
- $nGen$: Numero de generaciónes.
- $popSize$: Tamaño de la población
- $maxTrees$: Número máximo de árboles para cada individuo.

Variables:

- UV : Arreglo que representa el flujo óptico para la secuencia de entrada.
- $maxLevel$: Número máximo de niveles que un árbol puede tener.
- n : Contador de iteraciones para el número de generaciones.
- $pChromCross$: Probabilidad de cruce a nivel cromosoma.
- $pGenCross$: Probabilidad de cruce a nivel gen.
- $pChromMut$: Probabilidad de mutación a nivel cromosoma.
- $pGenMut$: Probablilidad de mutación a nivel gen.
- $newGen$: Arreglo que almacenará a la nueva población.
- $parents$: Arreglo que contiene los individuos seleccionados a los que se le aplicará algun operador genético.
- $offSpring$: Arreglo que almacena el resultado de aplicar un operador genético.

Salida: $bestInd$: Arreglo que representa al individuo con mejor desempeño.
Inicio

$UV \leftarrow calcOpFlow(Im1, Im2)$ /*Realiza la estimación de flujo óptico para la secuencia de entrada*/
$pop \leftarrow initPop(popSize, maxTrees, maxLevel, setFunctions, setTerminals)$ /*Inicializa la población*/
while $n < nGen$ **do**
    $evalPopFitness(pop, UV)$ /*Evaluación de la aptitud*/
    $newPop.clear()$
    while $newGen.length < popSize$ **do**
        $parents \leftarrow roulette(pop)$
        $operator \leftarrow rouletteOp(pChromCross, pGenCross, pChromeMut, pGenMut)$
        $offSpring \leftarrow applyOpe(operator, parents)$ /*Aplica operadores genéticos*/
        $newPop.add(offSpring)$
        $offSpring.clear()$
    end while
    $pop.replace(newPop)$ /*Reemplaza los individuos de la nueva población en la población original*/
    $bestInd \leftarrow getBest(pop)$ /*Obtiene el mejor individuo de la población*/
    $n = n + 1$
end while
return $bestInd$

El algoritmo anterior que va de la linea (1) a la (17), inicia con la asignación de la variable UV, que representa una estimación previa del flujo óptico para la secuencia de entrada utilizando el método de Lukas-Kanade en su variante piramidal abordado en el Capítulo 2. Además, $initPop$ es el método de inicialización de la población que en este caso corresponde al método *mitad y mitad*, requiere como parámetros de entrada el tamaño de la población (*popSize*), el número máximo de árboles que puede conformar a un individuo(*maxTrees*) así como el tamaño máximo de estos ($maxLevel$) .

A continuación, se evalúa el desempeño para cada individuo de la población inicial (4) y posteriormente se seleccionan aquellos que entrarán al proceso de síntesis por medio del método de la ruleta (7), basandosé en los valores de la aptitud obtenidos en el paso anterior: mientras este valor sea mayor entonces tendrá mayor probabilidad de ser seleccionado. Mediante el mismo método de selección por ruleta, se elige uno de los cuatro operadores genéticos para la etapa de variación (8), estos operadores son: cruce a nivel gen, cruce a nivel cromosoma, mutación a nivel gen y mutación a nivel cromosoma. Una vez elegido el operador, éste es aplicado (9) y después el resultado es añadido al arreglo $newPop$. Posteriormente este arreglo reemplaza a la generación anterior (13) y se almacena el mejor individuo de la generación en $bestInd$, este proceso se repite hasta alcanzar el número de generaciones previamente definido.

Algoritmo 9 initPop()

Descripción: Algoritmo que describe el proceso de inicialización de la población.
Entrada:

- $popSize$: Tamaño de la población
- $maxTrees$: Número máximo de árboles para cada individuo.
- $maxLevel$: Número máximo de niveles que un árbol puede tener.
- $setFunctions$: Arreglo que contiene el conjunto de funciones para generar los árboles
- $setTerminals$: Arreglo que contiene el conjunto de terminales para generar los árboles

Variables:

- $trees$: Número de árboles generado aleatoriamente.

Salida: pop : Población generada aleatoriamente
Inicio

1: **for** $i \leftarrow 1$ **to** $popSize$ **step** 1 **do**
2: $trees \leftarrow random(2, maxTrees)$ /*Genera un número aleatorio entre 2 y *maxTrees**/
3: $newInd \leftarrow 0$

```
    for t ← 1 to trees step 1 do
        newInd.tree(t) ← makeTree(setFunctions, setTerminals, maxLevel)
    end for
    pop(i) ← newInd
end for
return pop
```

El algoritmo crea una población basado en los parámetros de entrada. Se genera de manera aleatoria para cada individuo un número que representa la cantidad de árboles que lo compondrán, este varía entre dos y $maxTrees$.

Después cada árbol es generado de manera automática utilizando los arreglos *setTerminals* y *setFunctions*, que representan a los conjuntos de terminales y de funciones que son definidos previamente. Dichos conjuntos establecen el *Espacio de búsqueda* para el algoritmo evolutivo.

4.2.1.1. Funciones y terminales

En la Tabla 1, se muestran los conjuntos de funciones y terminales. El objetivo de los elementos que componen a estos dos conjuntos es que de manera automática construyan operadores que al ser sometidos al proceso de evolución y aplicados a la estimación del flujo óptico se mejore la precisión de ésta. A continuación se somete a evaluación el resultado obtenido usando la base de datos de Middlebury para flujo óptico cuyas métricas de evaluación se han descrito en la Sección 2.4.

Tabla 1. Funciones y terminales

$F = \{+, -, \times, \div, MF_3, MF_5, MF_7, log_2(), Dx(), Dy(), kTimes, kMinus, kPlus, kDiv\}$
$T = \{uv, Dx(uv), Dy(uv), Dxx(uv), Dyy(uv), MF_3, MF_5, MF_7\}$

En la Tabla 1, uv representa una estimación del flujo óptico. MF_s es un filtro de medianas y s indica el tamaño de este. D_u representa la derivada en la dirección de $u \in x, y, xx, yy$. Las primitivas $kTimes, kMinus, kPlus, kDiv$, realizan la operación que su nombre indica por una constante k, que es generada aleatoriamente durante el proceso de evolución.

4.2.1.2. Evaluación de aptitud

La aptitud de cada posible solución generada durante el proceso evolutivo se asigna evaluando la ecuación siguiente, donde M es el tamaño de la imagen, $I(x, y, t)$ la función de intensidad de la imagen en el punto (x, y) al tiempo t, las componentes del flujo óptico son u y v.

$$fitness = min\frac{1}{M}\Sigma_{i=1}^{M}|I_{ref}(x_i, y_i) - I(x_i + v_i, y_i + v_i)|$$

En el algoritmo propuesto se utiliza una estrategia de mapeo hacia atrás, mapeando los puntos de la imagen acorde al arreglo uv que corresponde a la estimación del flujo óptico a la que le han sido aplicados los operadores generados automáticamente por el algoritmo evolutivo. Se aplica la función de transformación al cuadro correspondiente a t_1 de la secuencia de entrada dando como resultado un nuevo cuadro en el que cada punto (x, y) del cuadro t_1 ha sido desplazado lo que indica las componentes u y v. Posteriormente se calcula la diferencia entre el cuadro de referencia t_2 de la secuencia de entrada y el cuadro obtenido en el paso anterior. Se considera mejor la aptitud del individuo evaluado mientras menor sea esta diferencia. A continuación se muestra el algoritmo de la función de evaluación.

Algoritmo 10 Proceso de evaluación de aptitud

Descripción: Algoritmo del proceso de evaluación de aptitud.
Entrada:

- ind : Arreglo que representa un individuo de la población.
- uv : Arreglo que representa la estimación previa del flujo.

Variables:

- $im1$: Arreglo que reprenta el cuadro en $t1$.
- $im2$: Arreglo que reprenta el cuadro en $t2$.
- nUV : Arreglo que almacena la nueva estimación del flujo.

Salida:

- $fitness$: Aptitud del individuo evaluado.

Inicio

1: **for each** *tree* **in** *ind.trees* **do**
2: $nUV \leftarrow nUV + eval(tree(uv))$
3: **end for each**
4: $remappedImg \leftarrow inverseMapping(im1, nUV)$
5: $fitness \leftarrow mean(abs(im2 - remappedImg))$
6: **return** $fitness$

4.2.1.3. Operadores genéticos

Aplicando el Algoritmo 8 después de creada la población inicial y evaluado el rendimiento de cada individuo, el siguiente paso corresponde a la selección y variación aplicando los correspondientes operadores genéticos como lo son *selección, cruce y mutación.*

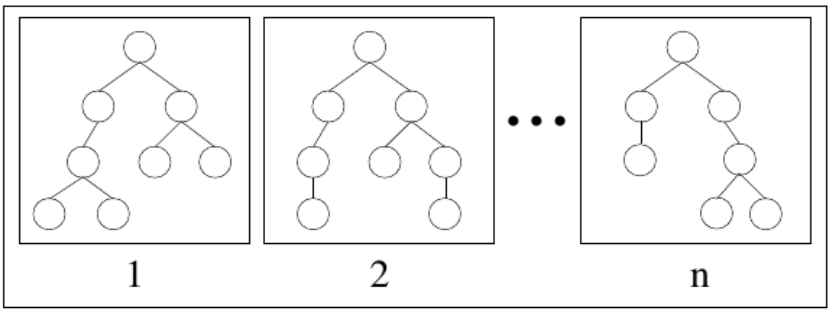


Figura 16. Estructura de un individuo

En la Figura 16 puede observarse un ejemplo de como está compuesto cada individuo de la población. Cada árbol t que forma parte de la estructura del inviduo está a su vez conformado por otras estructuras básicas conocidas como *nodos*. Debido a que un individuo consiste de un arreglo de árboles, los operadores genéticos definidos en el Capítulo 4 no pueden ser aplicados sin realizar antes algunas moficaciones.

Los operadores genéticos aplicables para el tipo de estructura que se tiene podemos clasificarlos como: nivel gen y nivel cromosoma. En las operaciones a nivel gen, tanto el cruce como la mutación se llevan a cabo dentro de la estructura de cada árbol que compone al individuo. El operador de cruce requiere un par de arreglos que representan a dos individuos padres necesarios para llevar a cabo el proceso de combinación y generar así dos nuevos individuos. Se genera de manera aleatoria un punto de cruce en cada uno de los árboles y es a partir de estos en los que se realiza el intercambio de información.

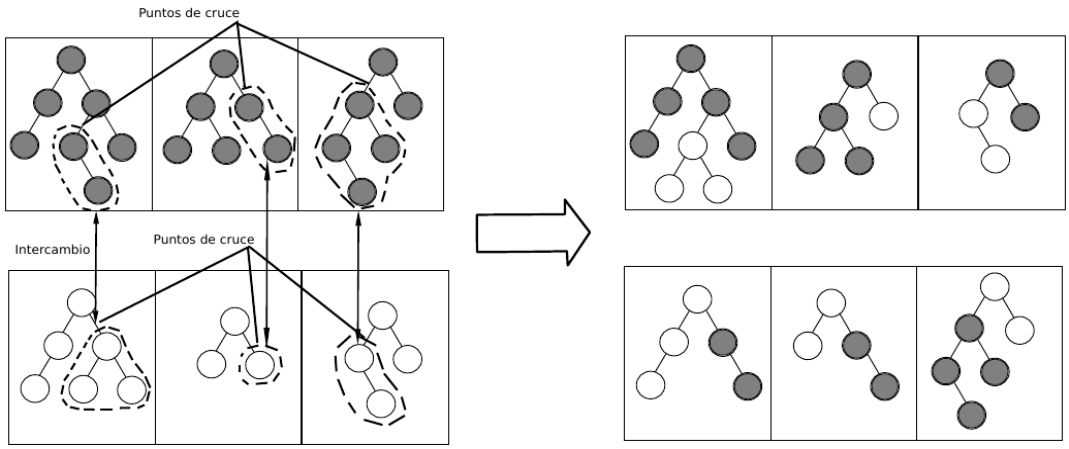


Figura 17. Cruce a nivel gen

Mientras que para la mutación es necesario solo un individuo, para cada árbol que lo compone se elige de manera aleatoria un nodo a partir del cual se aplicará el operador.

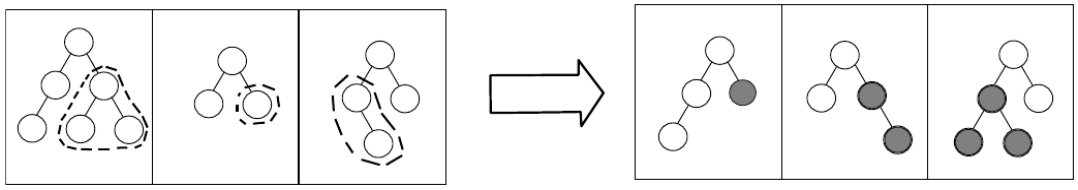


Figura 18. Cruce a nivel gen

Por otro lado, el cruce a nivel cromosoma consiste en el intercambio parcial de árboles de un individo padre a otro basado en un *punto de cruce* generado de manera aleatoria dando como resultado dos nuevos individuos.

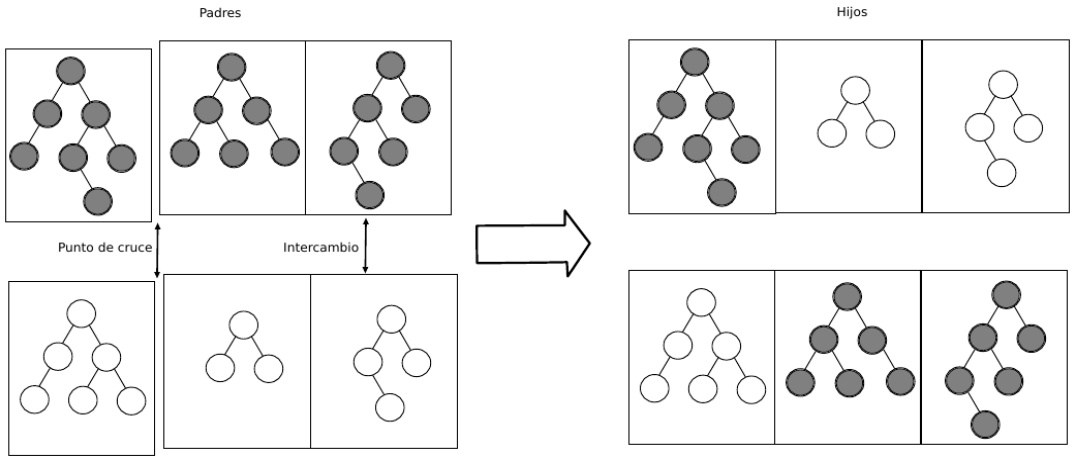


Figura 19. Cruce a nivel cromosoma

La mutación a nivel cromosoma consiste en cambiar por completo cada árbol que compone al individuo seleccionado dando como resultado uno nuevo, este operador solo requiere como entrada un solo individuo.

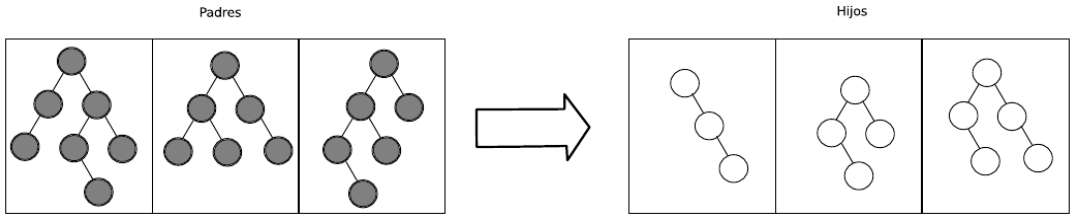


Figura 20. Mutación a nivel cromosoma

El algoritmo que describe el operador de cruce a nivel Gen, recibe como entrada dos arreglos que representan a dos individuos de la población($p1, p2$), que fueron elegidos previamente para entrar al proceso de combinación a nivel gen. En la línea (1) de entre los dos cromosomas de entrada se obtiene el tamaño del menor de ellos, esto con la finalidad de que el *punto de cruce* no exceda los límites de la estructura de menor tamaño.

El punto de cruce se genera de manera aleatoria (líneas 5 y 6) para cada uno de los cromosomas que conforma al individuo, este punto hace referencia a un gen (nodo) a partir del cual se realiza el intercambio de información (líneas 7 y 8) llevando consigo a los nodos que tenga como descendencia.

Algoritmo 11 Cruce a nivel Gen

Descripción: Algoritmo que describe el proceso cruce a nivel gen.
Entrada:

- $p1, p2$: Arreglos que representan a los cromosomas.

Variables:

- $minLen$: Indica el menor tamaño de entre los cromosomas de entrada.
- $cp1, cp2$: Puntos de cruce.

Salida: $c1, c2$: Arreglos resultantes del proceso de combinación a nivel gen.
Inicio

1: $minLen \leftarrow min(p1.length, p2.length)$
2: $c1 \leftarrow p1$
3: $c2 \leftarrow p2$
4: **for** $i \leftarrow 1$ **to** $minLen$ **step** 1 **do**
5: $\quad cp1 \leftarrow random(p1[i])$
6: $\quad cp2 \leftarrow random(p2[i])$
7: $\quad c1[i] \leftarrow swap(p1[i], p2[i], cp1, cp2)$
8: $\quad c2[i] \leftarrow swap(p2[i], p1[i], cp2, cp1)$
9: **end for**
10: **return** $c1, c2$

Algoritmo 12 Cruce de cromosoma

Descripción: Algoritmo que describe el proceso de cruce a nivel cromosoma.
Entrada:

- $p1, p2$: Arreglos que representan a los cromosomas.

Variables:

- $minLen$: Indica el menor tamaño de entre los cromosmas de entrada.
- cp : Punto de cruce.

Salida: $c1, c2$: Arreglos resultantes del proceso de combinación a nivel cromosoma.
Inicio

1: $minLen \leftarrow min(p1.length, p2.length)$
2: $c1 \leftarrow p1$
3: $c2 \leftarrow p2$
4: **if** $p1.length == p2.length$ **then**
5: $\quad minLen \leftarrow minLen - 1$
6: **end if**
7: $cp \leftarrow random(1, minLen)$
8: **for** $i \leftarrow 1$ **to** cp **step** 1 **do**
9: $\quad c1[i] \leftarrow p1[i]$
10: $\quad c2[i] \leftarrow p2[i]$
11: **end for**
12: **return** $c1, c2$

Al igual que el cruce a nivel gen, este algoritmo recibe como parámetro de entrada dos arreglos los cuales representan a los cromosomas. Se comienza por obtener de entre éstos el tamaño del menor de ellos (línea 1) y en caso de que ambos sean del mismo tamaño esta cantidad se disminuye en uno, ya que de lo contrario cuando se obtenga de manera aleatoria el punto de cruce existe la posibilidad de que éste sea igual a la longitud del cromosoma. Así se obtendría como resultado para este procedimiento un intercambio total de árboles es decir, la salida consistiría en una réplica de los arreglos de entrada. Una vez obtenido el punto de cruce se procede a realizar el intercambio de árboles que componen al individuo, desde la primera posición y teniendo como límite este punto (líneas 8-11). Por último los nuevos cromosomas son retornados.

Algoritmo 13 Mutación a nivel gen

Descripción: Algoritmo que describe el proceso mutación a nivel gen.
Entrada:

- p : Arreglo que representa un cromosoma al que será aplicado el operador.
- $maxLevel$: Tamaño máximo que puede tener el cromosoma.

Variables:

- mp : Número aleatorio que indica el nodo a partir del cual el operador será aplicado.
- $mpLvl$: Indica el nivel en el cual se encuentra el nodo al que apunta mp
- new : Arreglo que almacena la nueva rama.

Salida: c : Arreglo resultante del proceso de mutación a nivel gen.
Inicio

1: **for** $i \leftarrow 1$ **to** $p.length$ **step** 1 **do**
2: $\quad mp \leftarrow random(p[i])$
3: $\quad mpLlv \leftarrow getNodeLvl(mp)$
4: $\quad new \leftarrow makeTree(setFunctions, setTerminals, maxLevel - mpLvl + 1)$
5: $\quad c[i] \leftarrow joinTrees(p[i], new, mp)$
6: **end for**
7: **return** c

El algoritmo recibe como entrada un arreglo que representa un individuo. Para cada árbol que compone al individuo se selecciona de manera aleatoria un nodo a partir del cual se aplicará el operador (línea 2). Después se obtiene el nivel en el cual se encuentra dicho nodo, con la finalidad de que al añadir nueva información al árbol a partir de este nodo (línea 4) no se supere el tamaño máximo que puede tener cada árbol definido en la variable *maxLevel*. Por último, se añade esta nueva información al árbol (línea 5) y se repite el procedimiento con todos los árboles que componen al individuo.

Algoritmo 14 Mutación de cromosoma

Descripción: Algoritmo que describe el proceso mutación de cromosoma.
Entrada:

- p : Arreglo que representa un cromosoma al que será aplicado el operador.
- $maxLevel$: Tamaño máximo que puede tener el cromosoma.

Variables:

- new : Arreglo que almacena el nuevo cromosoma.

Salida: c : Arreglo resultante del proceso de mutación a nivel cromosoma.
Inicio

1: **for** $i \leftarrow 1$ **to** $p.length$ **step** 1 **do**
2: $new \leftarrow makeTree(setFuncions, setTerminals, maxLevel)$
3: $c[i] \leftarrow new$
4: **end for**
5: **return** c

Este algoritmo es básico en su funcionamiento, ya que recibe como entrada un arreglo que representa a un individuo y cada uno de los árboles que lo componen son reemplazados por unos nuevos que son generados de manera aleatoria dado el conjunto de funciones y terminales, respetando los límites de tamaño máximo definidos en *maxLevel*.

4.2.2. Segunda propuesta

En esta segunda versión del método propuesto se integra la información que proveen las secuencias de imágenes a color. Para esto se definió una nueva estructura para los individuos que ahora se componen de cinco árboles. El primero se conforma de operaciones que actúan sobre el espacio de color para la secuencia evaluada, con el fin de resaltar características que ayuden realizar una mejor estimación del flujo. Los tres siguientes árboles actúan sobre la estimación previa del flujo óptico con la finalidad de realizar correciones en la estimación. La función del árbol restante es integrar de alguna manera los tres árboles anteriores que forman los operadores que actúan sobre el flujo. Se definieron nuevos conjuntos de funciones y terminales que forman la nueva estructura de los individuos.

Tabla 2. Funciones y terminales para la dimensión de color

$F = \{+, -, \times, \div, MF_3, MF_5, MF_7, GaussF_{\sigma=1}, GaussF_{\sigma=2}\}$
$T = \{R, G, B, H, S, V, S_{HSI}, I, Y, U, V_{YUV}\}$

en donde MF_n es un filtro de medianas de tamaño n, $GaussF_{\sigma=i}$ un filtro Gaussiano con valor de sigma i . El conjunto de terminales está integrado por bandas que componen cada secuencia en los modelos de color RGB, HSV, HSI, YUV.

Tabla 3. Funciones y terminales para la corrección de flujo

$F = \{+, -, \times, \div, MF_3, MF_5, MF_7, GaussF_{\sigma=1}, GaussF_{\sigma=2}, Dx(), Dy()$ $kTimes, kMinus, kPlus, kDiv\}$
$T = \{uv, Dx(uv), Dy(uv), MF_3(uv), MF_5(uv), MF_7(uv), GaussF_{\sigma=1}(uv), GaussF_{\sigma=2}(uv)\}$

D_x, D_y son las derivadas parciales de la imagen en la dirección de x y y respectivamente. $kTimes, kMinus, kPlus, kDiv$ son operaciones de multiplicación, resta, suma y división por una constante k.

Tabla 4. Funciones y terminales para la función de integración de operadores

$F = \{+, -, \times, \div, MF_3, MF_5, MF_7, GaussF_{\sigma=1}, GaussF_{\sigma=2}, Dx(), Dy()$ $kTimes, kMinus, kPlus, kDiv\}$
$T = \{Op1, Op2, Op3\}$

$Op1, Op2, Op3$ representan a los operadores generados de manera automática para la corrección del flujo, los cuales son combinados de acuerdo a la estructura generada por este árbol. Debido a la nueva estructura de los individuos el proceso de inicialización de la población así como la evaluación cambia. Estos se describen a continuación en el Algoritmo 15 y Algoritmo 16 respectivamente.

Algoritmo 15 initPop() — B

Descripción: Algoritmo que describe el proceso de inicialización de la población.
Entrada:

- $popSize$: Tamaño de la población
- $maxTrees = 5$: Número máximo de árboles para cada individuo.
- $maxLevel$: Número máximo de niveles que un árbol puede tener.
- $setFunctionsCol$: Arreglo que contiene el conjunto de funciones para la dimensión de color.
- $setTerminalsCol$: Arreglo que contiene el conjunto de terminales para la dimensión de color.
- $setFunctionsFlw$: Arreglo que contiene el conjunto de funciones para generar operadores de corrección de flujo.
- $setTerminalsFlw$: Arreglo que contiene el conjunto de terminales para generar operadores de corrección de flujo.
- $setFunctionsFI$: Arreglo que contiene el conjunto de funciones para la integración de operadores de corrección de flujo.
- $setTerminalsFI$: Arreglo que contiene el conjunto de terminales parala integración de operadores de corrección de flujo.

Variables:

- $trees$: Número de árboles que componen al individuo.

Salida: pop : Población generada aleatoriamente
Inicio

1: **for** $i \leftarrow 1$ **to** $popSize$ **step** 1 **do**
2: $trees \leftarrow 5$
3: $newInd \leftarrow 0$
4: $newInd.tree(1) \leftarrow makeTree(setFunctionsCol, setTerminalsCol, maxLevel)$
5: **for** $t \leftarrow 2$ **to** 4 **step** 1 **do**
6: $newInd.tree(t) \leftarrow makeTree(setFunctionsFlw, setTerminalsFlw, maxLevel)$
7: **end for**
8: $newInd.tree(5) \leftarrow makeTree(setFunctionsFI, setTerminalsFI, maxLevel)$
9: $pop(i) \leftarrow newInd$
10: **end for**
11: **return** pop

Algoritmo 16 Evaluación de aptitud — B

Descripción: Algoritmo del proceso de evaluación de aptitud.
Entrada:

- ind : Arreglo que representa un individuo de la población.

Variables:

- im : Arreglo que almacena la secuencia de imágenes.
- $im1$: Arreglo que representa $t1$ de la secuencia.
- $im2$: Arreglo que representa $t2$ de la secuencia.
- flw : Arreglo que almacena una estimación del flujo.
- uv : Arreglo que almacena el flujo óptico corregido.

Salida:

- $fitness$: Aptitud del inviduo evaluado.

Inicio

1: $imgs \leftarrow eval(ind(1), im)$
2: $flw \leftarrow LKP(imgs)$
3: $Op1 \leftarrow eval(ind(2))$
4: $Op2 \leftarrow eval(ind(3))$
5: $Op3 \leftarrow eval(ind(4))$
6: $uv \leftarrow eval(Op1, Op2, Op3, uv)$
7: $remappedImg \leftarrow inverseMapping(im1, uv)$
8: $fitness \leftarrow mean(abs(im2 - remappedImg))$
9: **return** $fitness$

4.3. Implementación

La implementación de este algoritmo se realizó en el entorno MATLAB utilizando el toolbox para programación genética GPLAB.

Además se utilizó para la evaluación la base de datos de Middlebury para flujo óptico, que se compone de dos conjuntos de doce secuencias con un máximo de ocho cuadros cada una. Uno de estos conjuntos poseé valores de referencia para los vectores de flujo *"ground-truth"* público para ocho de las secuencias el cual es usado como conjunto de entrenamiento, mientras que el otro conjunto es el de prueba y su *"ground-truth"* no está disponible públicamente.

Capítulo 5. Experimentos y resultados

5.1. Experimentos realizados

Parte de los primeros experimentos realizados consistió en evaluar ocho de las secuencias de la base de datos de Middlebury para flujo óptico, que se muestran en la Figura 21 utilizando los métodos de Horn-Schunck (HS) y Lucas-Kanade, en su enfoque clasico y piramidal (LK y PLK). Los resultados se muestran a continuación en la Tabla 5, en donde se reporta el promedio de los errores de punto final (*EPE*) y el error angular promedio (*AAE*) para el flujo óptico estimado en cada una de las secuencias. Estos errores se explican en la Sección 2.4, mientras menor sean ambos errores se considera mayor la precisión en la estimación del flujo óptico.

Tabla 5. Tabla comparativa — Métodos de estimación de flujo óptico

Método	Dimetrodon		Grove2		Grove3		Hydrangea		RubberWhale		Urban2		Urban3		Venus	
	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE
HS	53.844	2.324	74.179	4.510	68.240	5.828	66.507	4.234	32.191	1.167	65.283	9.392	71.772	8.528	70.734	5.047
LK	50.109	1.831	67.814	3.107	61.748	3.974	61.860	3.552	22.312	0.709	62.322	8.609	71.077	7.822	64.986	3.862
PLK	13.472	0.637	9.536	0.689	18.332	1.908	10.357	0.895	14.215	0.477	28.464	6.004	34.318	5.430	22.328	1.678

Con la finalidad de tener un punto de referencia con el cúal poder comparar los resultados obtenidos por los métodos propuestos en este trabajo, se considera el método *Classic+NL* descrito en (Sun, 2010), al momento de su publicación se encontraba clasificado en el primer puesto del benchmark de Middlebury para evaluación del flujo óptico. En dicho trabajo se realizó un estudio de diferentes métodos para la estimación de flujo óptico considerando diferentes practicas referentes al modelado del problema y su implementación e identificando aquellos aspectos que influyen para una estimación precisa del flujo óptico.

Se elegió este método debido a que el código de la implementación en MATLAB se encuentra disponible, dando así la posibilidad de evaluar las secuencias del conjunto de entrenamiento y comparar los resultados obtenidos por los métodos propuestos en este trabajo. Si bien este método actualmente ya no se encuentra posicionado en primer lugar, brinda un desempeño comparable con aquellos métodos que se posicionan en los primeros lugares del benchmark de Middlebury, evaluando el conjunto de secuencias de prueba cuyos valores de referencia no se encuentra disponible de manera pública.

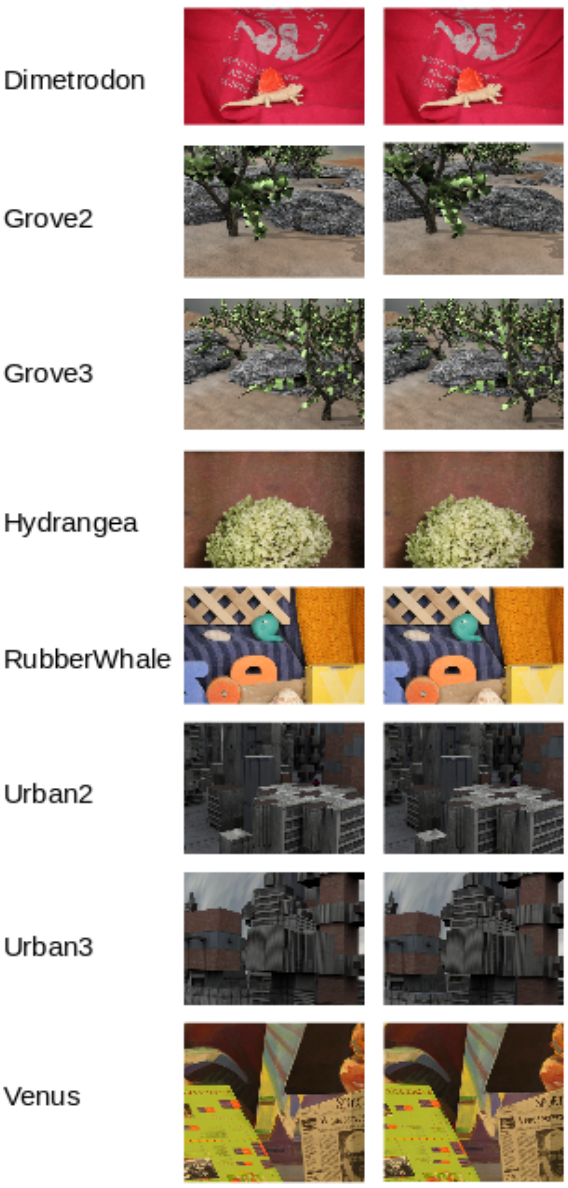


Figura 21. Secuencias evaluadas

A partir de ahora se hará referencia al método de Lucas-Kanade en su enfoque piramidal modificado por el programa de GP como *"gpPLK"* y *"gpPLKColor"* para el método propuesto en su segunda versión que integra información de color en las secuencias de imágenes.

Todos los experimentos fueron realizados en cuatro estaciones de trabajo usando como sistema operativo GNU/Linux openSUSE en sus versiones 13.1 y 42.2 y MATLAB®R2011. Cada equipo cuenta con las siguientes caracteristicas:

- Dell precision T7600 con una arquitectura x86-64, procesador Intel®Xeon E5-2609 2.40Ghz de 8 núcleos, 8 GB de memoria RAM, Tarjeta gráfica NVIDIA®GF100GL Quadro 4000.

En la Tabla 6 se muestran los resultados obtenidos al ejecutar el programa de GP evaluando las 8 secuencias del conjunto de entrenamiento de la base de datos de Middlebury, mediante los métodos propuestos en este trabajo. Como se puede observar el método que incorpora información a color presenta un mejor desempeño, arrojando un error menor en la estimación del flujo óptico para cada una de las secuencias evaluadas. Adémas como referencia se incluyen los resultados obtenidos mediante el método *Classic+NL* evaluando el mismo conjunto de imágenes.

Tabla 6. Tabla comparativa — resultados preliminares

Método	Dimetrodon		Grove2		Grove3		Hydrangea		RubberWhale		Urban2		Urban3		Venus	
	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE
Classic+NL	2.280	0.117	1.410	0.098	4.927	0.464	1.824	0.151	2.401	0.076	2.034	0.210	3.160	0.421	3.289	0.232
gpPLK	11.2552	0.5514	6.5765	0.4513	13.5088	1.4215	6.3062	0.5769	11.1924	0.3565	23.8622	5.4499	31.2206	5.0296	19.0584	1.4211
gpPLKColor	3.4191	0.1696	4.4269	0.3033	11.579	1.2465	5.1464	0.4610	8.2622	0.2841	23.5533	5.407	26.1154	5.0262	12.4474	0.9174

Por cada ejecución del programa de GP evaluando una secuencia de imágenes se obtuvó un individuo diferente, por lo que acontinuación se realizaron experimentos para encontrar si existia uno o varios individuos que brindaran un mejor desempeño en la estimación del flujo óptico para todas las secuencias. Para estos experimentos se consideraron únicamente los mejores individuos obtenidos mediante el programa de GP que incorpora información a color, ya que este arrojó mejores resultados. Se evaluó cada individuo en cada secuencia del conjunto de entrenamiento y los resultados obtenidos se muestran en la Tabla 7. Los errores de las mejores estimaciones se resaltan en *negritas*, como se puede observar es el *individuo 8* aquel que genera una estimación más precisa para la mayoria de las secuencias.

Tabla 7. Evaluación de los mejores individuos obtenidos en todas las secuencias

	Dimetrodon		Grove2		Grove3		Hydrangea		RubberWhale		Urban2		Urban3		Venus	
	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE
Ind_1	3.4191	0.16863	4.4004	0.29667	12.1269	1.2778	5.1899	0.46285	7.8927	0.25054	21.7879	5.4115	22.8115	4.0818	14.2438	1.0888
Ind_2	6.0041	0.28759	4.4269	0.30338	**11.4677**	**1.2317**	5.4531	0.50809	7.8381	0.25405	22.9376	5.579	22.6798	4.22	21.5273	1.5663
Ind_3	6.2188	0.29666	4.4767	0.30787	11.579	1.2465	5.5991	0.52197	7.9436	0.25916	23.355	5.6557	23.1165	4.306	21.9941	1.6142
Ind_4	8.519	0.42414	5.7371	0.38999	13.1861	1.3749	5.1464	0.46108	13.8495	0.42172	23.6343	5.4142	28.8808	4.7825	16.6127	1.2066
Ind_5	4.9762	0.24532	5.4629	0.40488	13.4555	1.4616	5.6851	0.60006	8.2622	0.2841	28.7582	7.2032	27.4422	7.5229	17.2754	1.4307
Ind_6	8.5197	0.42432	5.7366	0.38985	13.1543	1.3707	5.1416	0.46007	13.8441	0.4217	23.5533	5.407	28.7921	4.7672	16.5913	1.2036
Ind_7	8.6392	0.41093	5.3281	0.38447	13.0381	1.4335	6.6528	0.6404	9.299	0.3264	26.3486	6.2167	26.1154	5.0262	25.9576	2.0484
Ind_8	**2.9467**	**0.15179**	**4.2432**	**0.28208**	11.6503	1.2218	**4.5877**	**0.41167**	**7.5257**	**0.23112**	**19.3693**	**5.0661**	**20.8328**	**3.6875**	**12.4474**	**0.91747**

En la siguiente tabla se muestran resaltados en negritas los mejores resultados obtenidos mediante el método propuesto, asi como los resultados obtenidos con el método Classic+NL y PLK, que es el método que se usó como base para el método propuesto.

Tabla 8. Tabla comparativa—resultados finales

Método	Dimetrodon		Grove2		Grove3		Hydrangea		RubberWhale		Urban2		Urban3		Venus	
	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE	AAE	EPE
Classic+NL	2.280	0.117	1.410	0.098	4.927	0.464	1.824	0.151	2.401	0.076	2.034	0.210	3.160	0.421	3.289	0.232
PLK	13.472	0.637	9.536	0.689	18.332	1.908	10.357	0.895	14.215	0.477	28.464	6.004	34.318	5.430	22.328	1.678
gpPLKColor	**2.9467**	**0.1517**	**4.2432**	**0.2820**	**11.4677**	**1.2317**	**4.5877**	**0.4116**	**7.5257**	**0.2311**	**19.3693**	**5.0661**	**20.8328**	**3.6875**	**12.4474**	**0.9174**

En la Sección 5.1.1 se muestran las gráficas del proceso evolutivo para el método *gpPLK*, mientras que para el método *gpPLK-Color* se muestran en la Sección 5.1.2. Los resultados obtenidos son los que se reportan en la Tabla 6. Posteriormente en la sección se puede visualizar una tabla comparativa con el flujo óptico estimado para cada una de las secuencias mediante las dos variantes del método propuesto, así como generado por el mejor individuo obtenido.

5.1.1. gpPLK

Los parámetros proporcionados al programa de GP se muestran en la siguiente tabla.

Tabla 9. Parametros GP

Número de ejecuciones	30	
Tamaño de la población	30	
Método de inicialización	Ramped Half-and-Half	
Número de generaciónes	30	
Número máximo de árboles por individuo	6	
Profundidad de los árboles	5	
Profundidad máxima dinámica	7	
Profundidad máxima real	9	
Probabilidades de operadores genéticos	Cruce nivel gen. = 0.8	Mutanción nivel gen. = 0.8
	Mut. nivel gen. = 0.2	Mut. nivel gen. = 0.2
Selección	Torneo Lexicográfico	
Supervivencia	Reemplazo generacional	

A continuación se muestran los gráficos con las estadísticas del proceso evolutivo para cada una de las secuencias evaluadas con el método propuesto en su primera versión.

5.1.1.1. Resultados de la secuencia Dimetrodon

La gráfica de la Figura 22 corresponde a la evolución promedio de 30 ejecuciones del programa de GP evaluando la secuencia Dimetrodon. Se observa que los mejores individuos obtuvieron una aptitud de 1.7605. La aptitud máxima que puede obtener un individuo al ser evaluado para todos los casos es 0. El error angular promedio en la estimación del flujo para esta secuencia fué de 11.2592, el error de punto final de 0.5514.

Por último la Figura 25 muestra el flujo óptico estimado, en las formas de representación descritas en la Sección 2.2.5. Este orden de aparición de figuras se repite con el resto de las secuencias evaludas.

La Figura 23 muestra la evolución que se tuvo en la ejecución del programa de GP en donde se encontró al individuo con mejor aptitud de las 30 ejecuciones. En la Figura 24 se muestra la frecuencia de uso de las funciones empleadas para la contrucción de los individuos. Debido a que el conjunto de terminales descrito en la Tabla 1 consiste en el resultado de haber sido aplicado

una función del conjunto de funciones, esta gráfica se omite. Este conteo se realiza considerando únicamente los 30 mejores individuos del total de ejecuciones, es decir el mejor de cada ejecución.

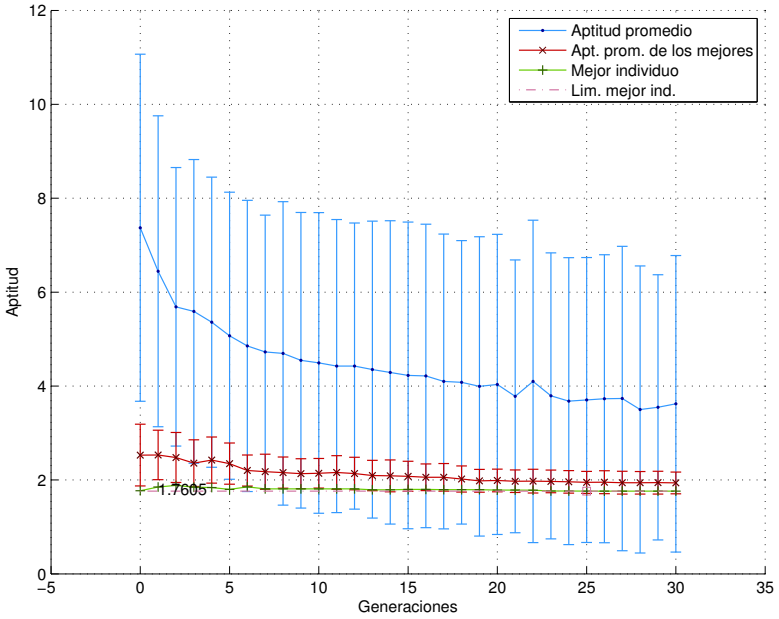


Figura 22. Evolución promedio de 30 ejecuciones del programa de GP para la secuencia Dimetrodon

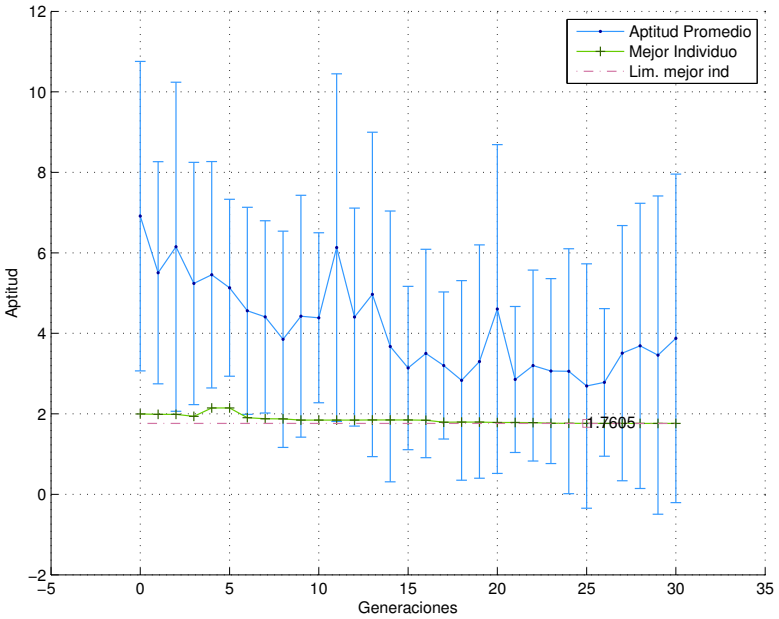


Figura 23. Evolución de la mejor ejecución para la secuencia Dimetrodon

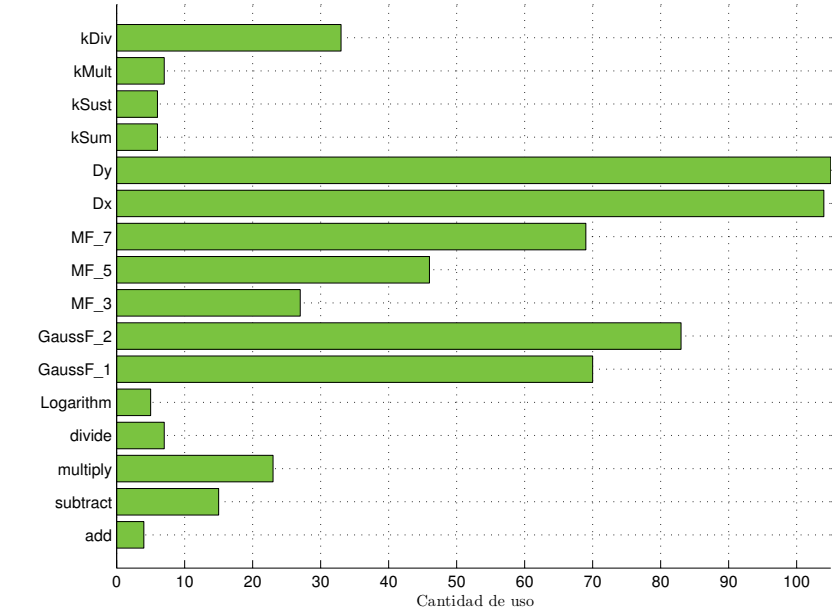


Figura 24. Frecuencia de uso de funciones para la secuencia Dimetrodon

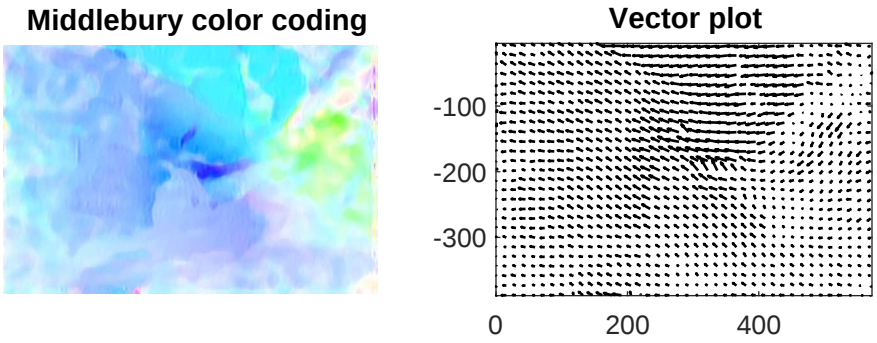


Figura 25. Flujo estimado para la secuencia Dimetrodon — AAE 11.2592 EPE 0.5514

5.1.1.2. Resultados de la secuencia Groove2

Para esta secuencia la aptitud del mejor individuo obtenida fue de 5.0632. El error angular promedio en la estimación del flujo para esta secuencia fué de 6.5765, el error de punto final de 0.4513.

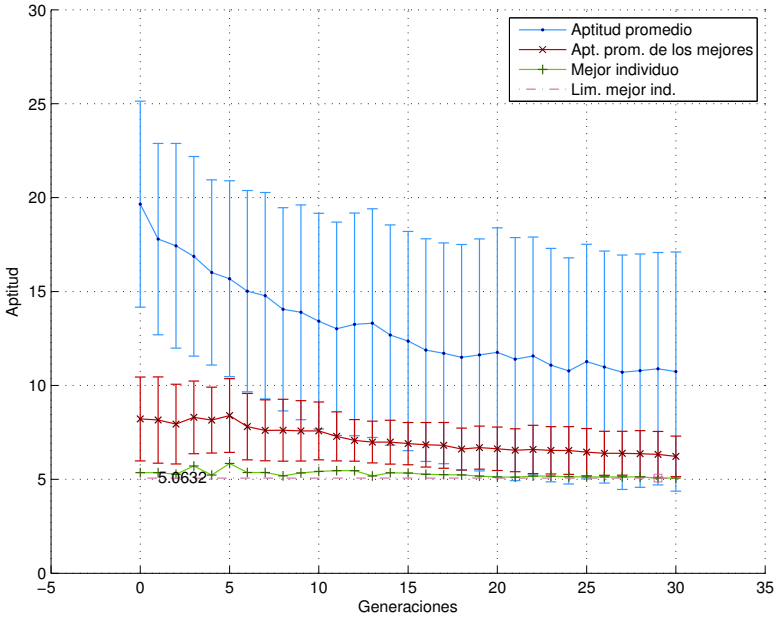


Figura 26. Evolución promedio de 30 ejecuciones del programa de GP para la secuencia Groove2

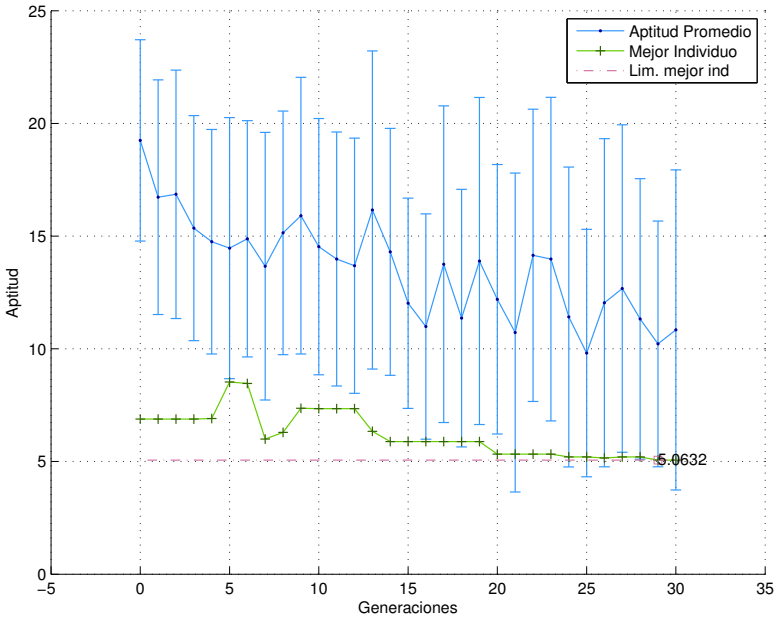


Figura 27. Evolución de la mejor ejecución para la secuencia Groove2

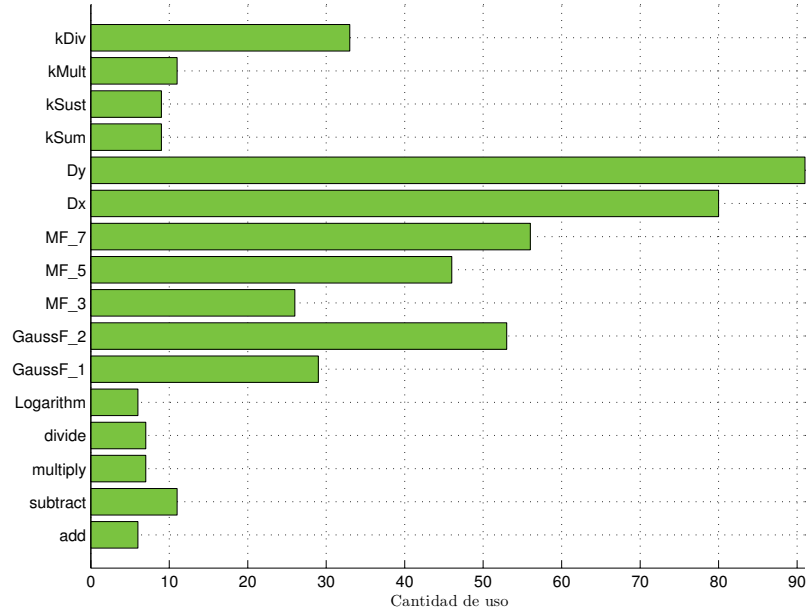


Figura 28. Frecuencia de uso de funciones para la secuencia Groove2

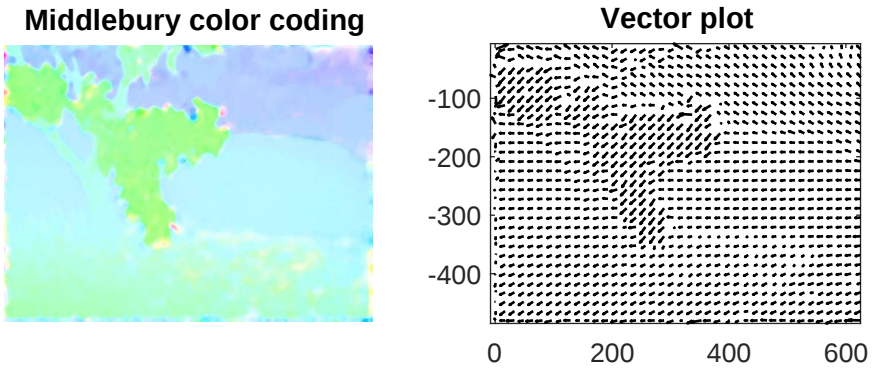


Figura 29. Flujo estimado para la secuencia Groove2 — AAE 6.5765 EPE 0.4513

5.1.1.3. Resultados para la secuencia Groove3

Para esta secuencia la aptitud del mejor individuo obtenida fue de 8.2619. El error angular promedio en la estimación del flujo para esta secuencia fué de 13.5088, mientras que el error de punto final de 1.4215.

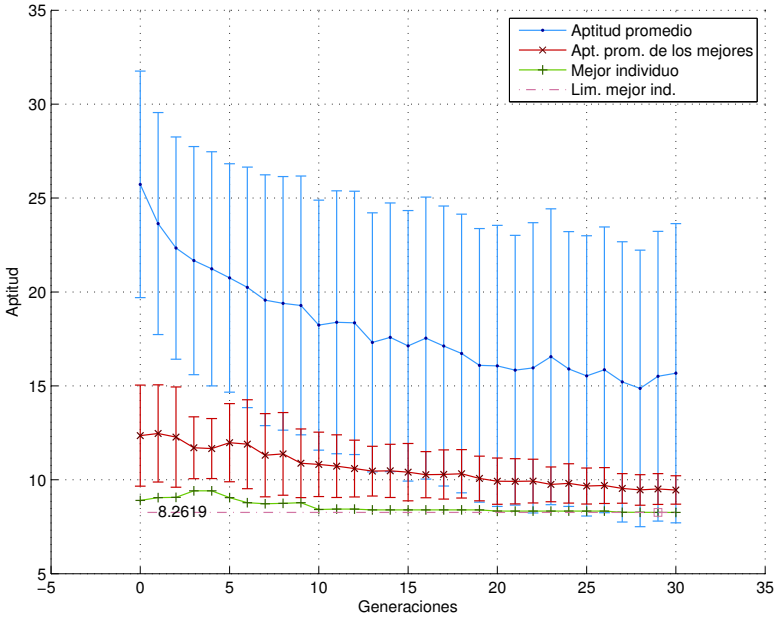


Figura 30. Evolución promedio de 30 ejecuciones del programa de GP para la secuencia Groove3

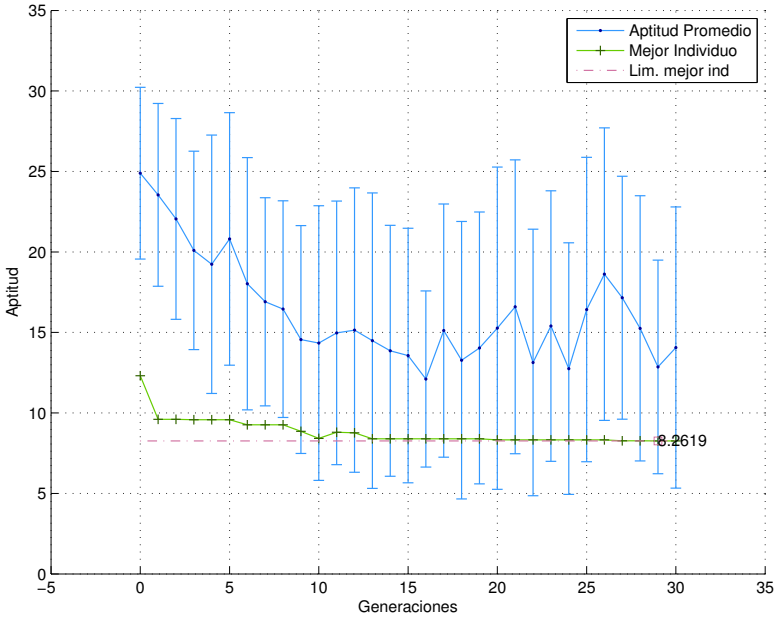


Figura 31. Evolución de la mejor ejecución para la secuencia Groove3

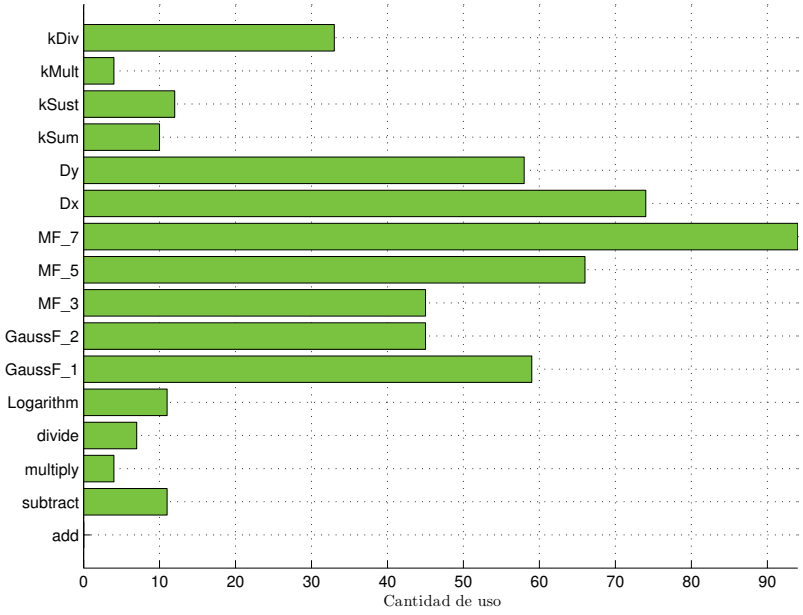


Figura 32. Frecuencia de uso de funciones para la secuencia Groove2

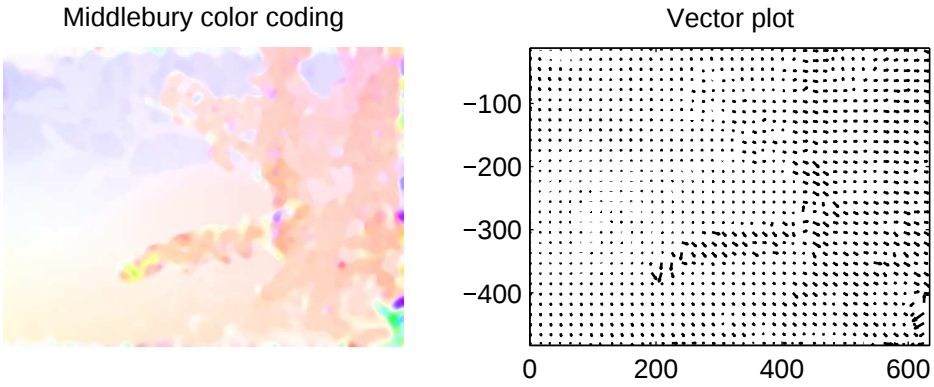


Figura 33. Flujo estimado para la secuencia Groove3 — AAE 13.5088 EPE 1.4215

5.1.1.4. Resultados de la secuencia Hydrangea

Para esta secuencia la aptitud del mejor individuo obtenida fue de 3.3627. El error angular promedio en la estimación del flujo para esta secuencia fué de 6.3062, mientras que el error de punto final de 0.5769.

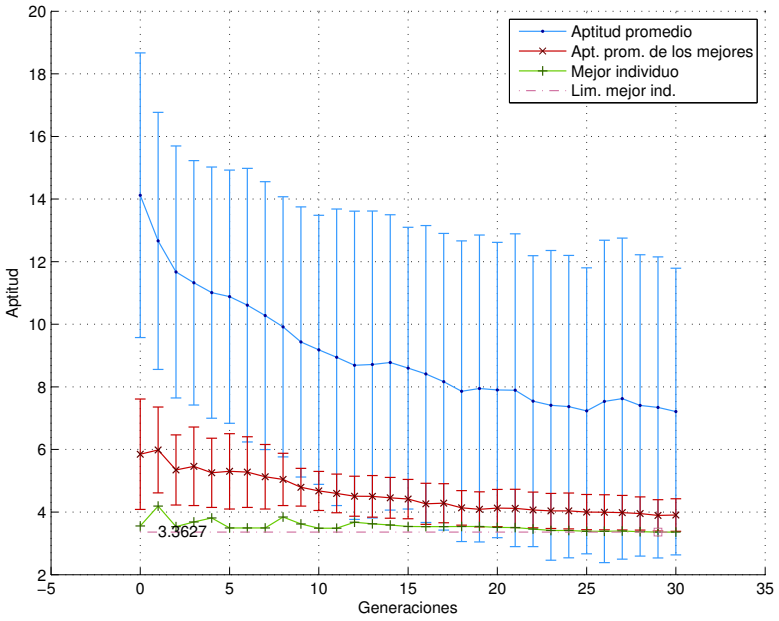


Figura 34. Evolución promedio de 30 ejecuciones del programa de GP para la secuencia Hydrangea

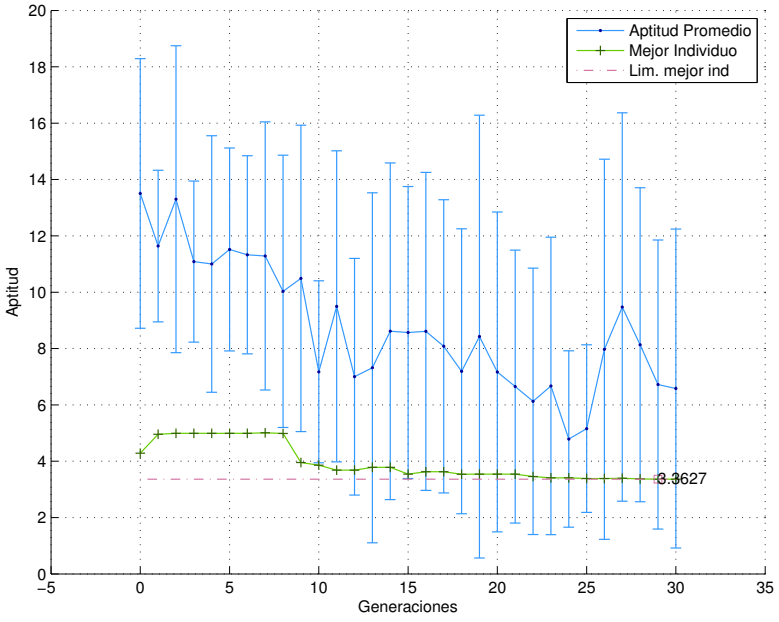


Figura 35. Evolución de la mejor ejecución para la secuencia Hydrangea

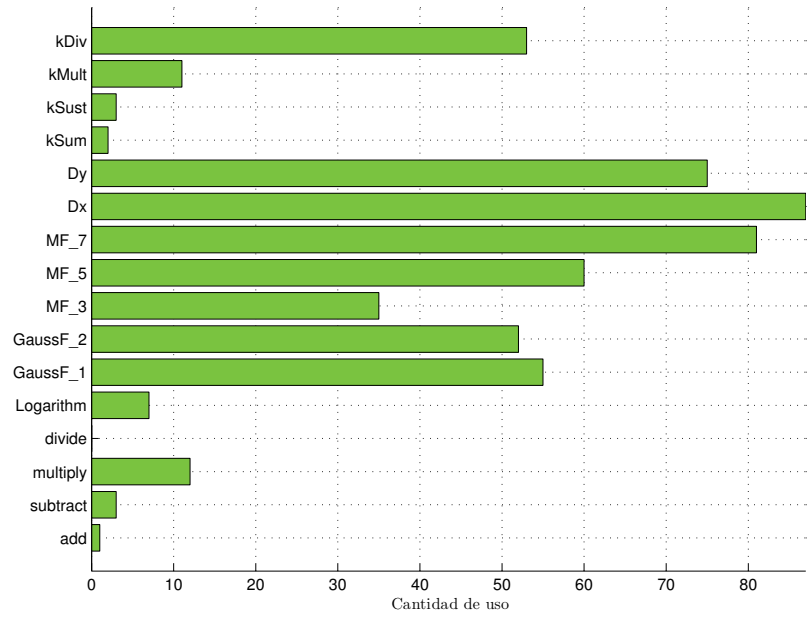


Figura 36. Frecuencia de uso de funciones para la secuencia Hydrangea

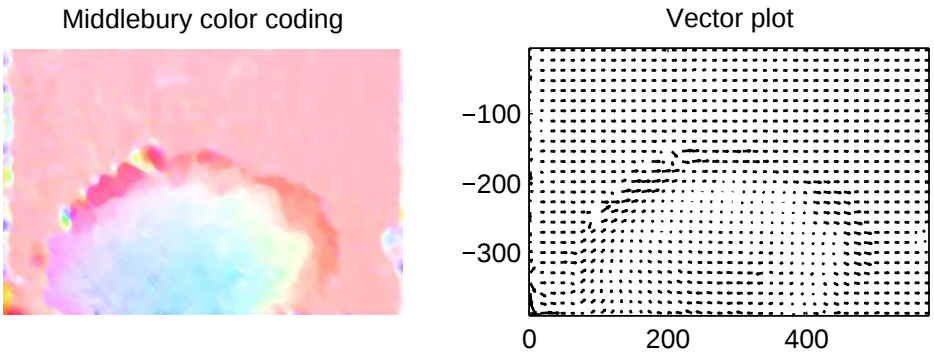


Figura 37. Flujo estimado para la secuencia Hydrangea — AAE 6.3062 EPE 0.5769

5.1.1.5. Resultados de la secuencia RubberWhale

Para esta secuencia la aptitud del mejor individuo obtenida fue de 1.424. El error angular promedio en la estimación del flujo para esta secuencia fué de 11.1924, mientras que el error de punto final de 0.3565.

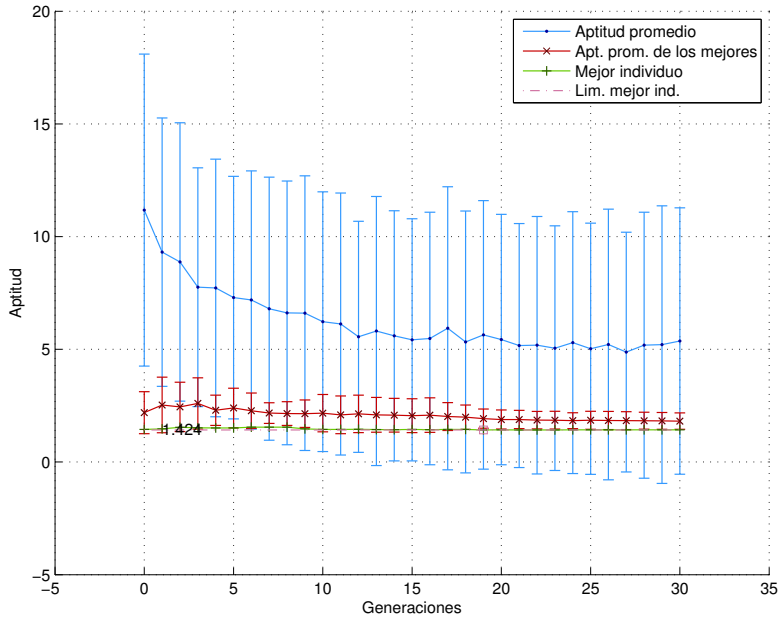


Figura 38. Evolución promedio de 30 ejecuciones del programa de GP para la secuencia RubberWhale

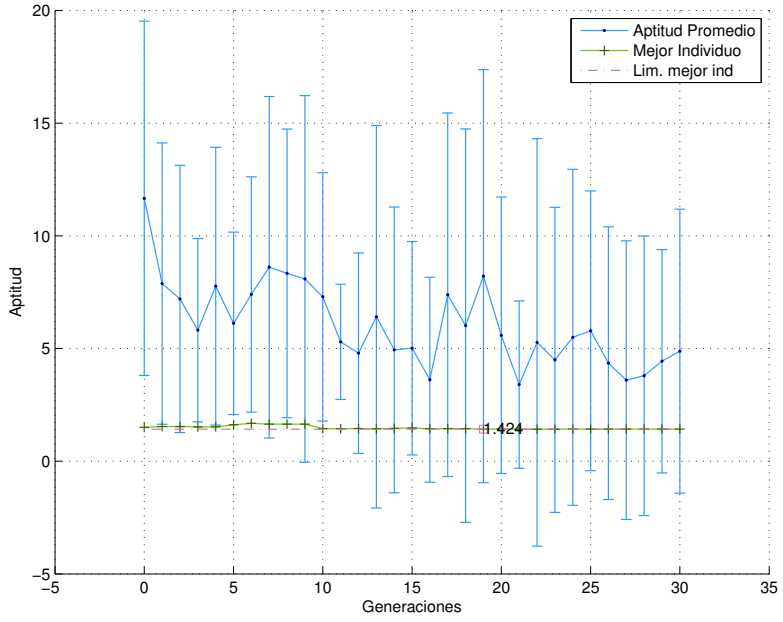


Figura 39. Evolución de la mejor ejecución para la secuencia RubberWhale

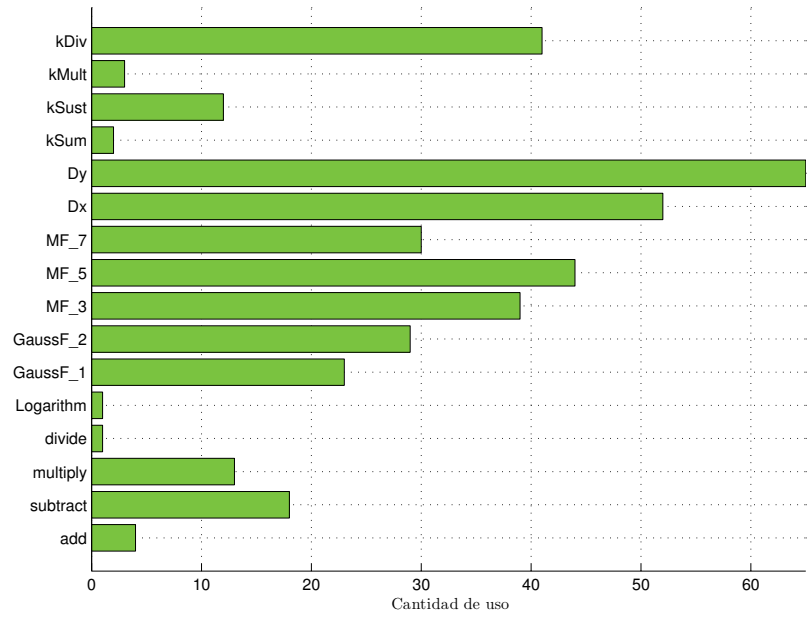


Figura 40. Frecuencia de uso de funciones para la secuencia RubberWhale

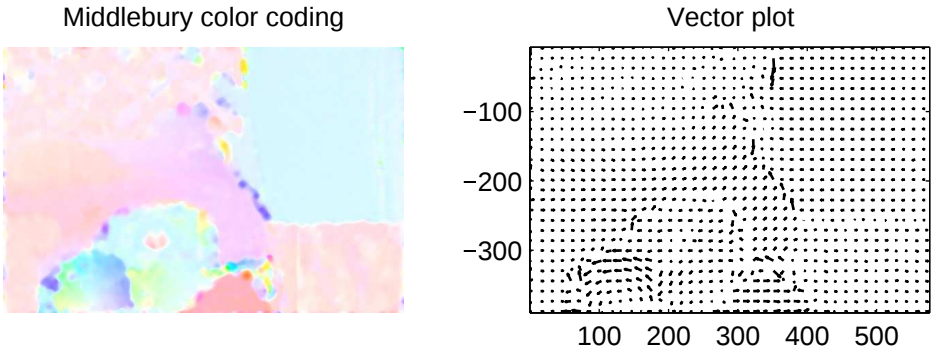


Figura 41. Flujo estimado para la secuencia RubberWhale — AAE 11.1924 EPE 0.3565

5.1.1.6. Resultados de la secuencia Urban2

Para esta secuencia la aptitud del mejor individuo obtenida fue de 5.8212. El error angular promedio en la estimación del flujo para esta secuencia fué de 23.8622, mientras que el error de punto final de 5.4499.

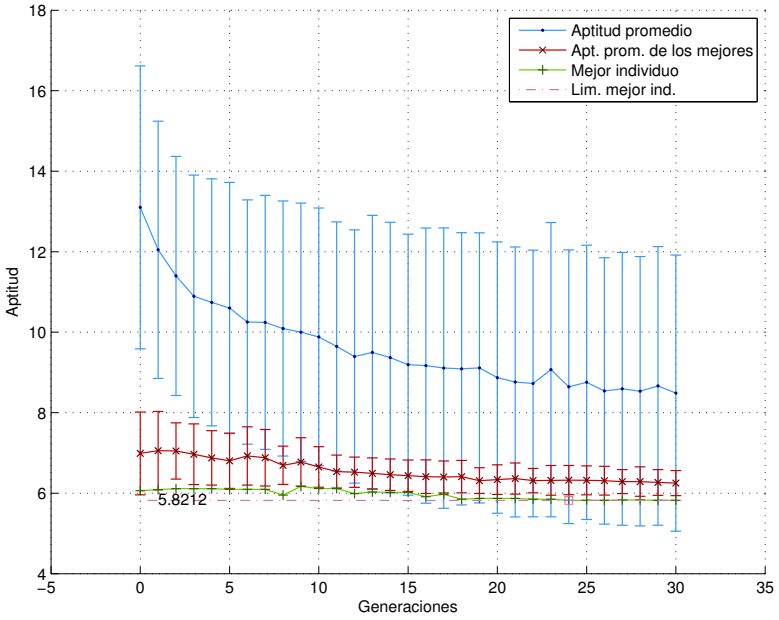


Figura 42. Evolución promedio de 30 ejecuciones del programa de GP para la secuencia Urban2

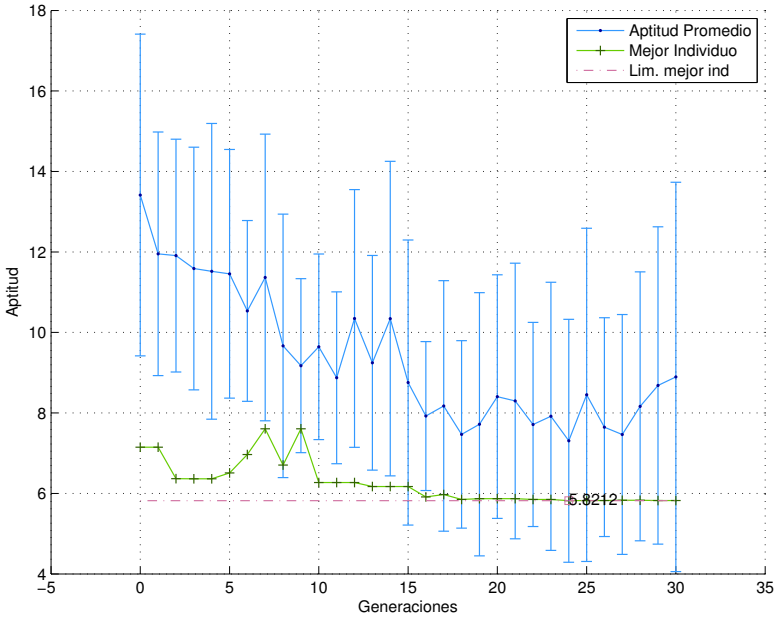


Figura 43. Evolución de la mejor ejecución para la secuencia Urban2

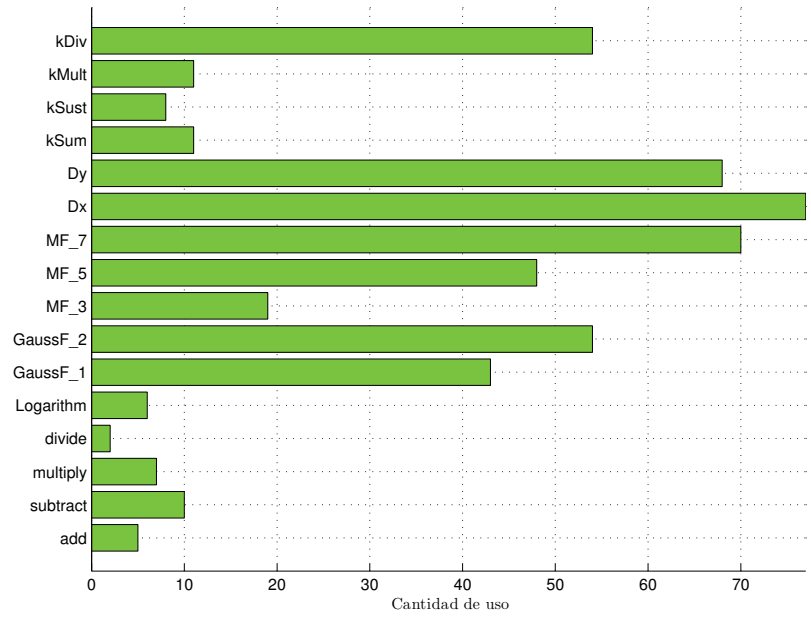


Figura 44. Frecuencia de uso de funciones para la secuencia Urban2

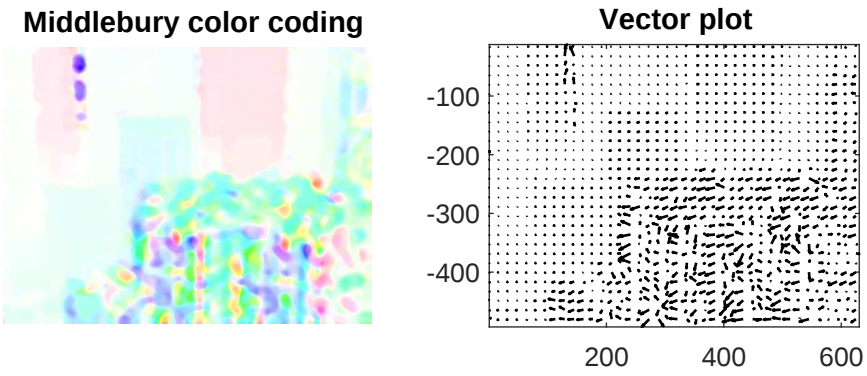


Figura 45. Flujo estimado para la secuencia Urban2 — AAE 23.8622 EPE 5.4499

5.1.1.7. Resultados de la secuencia Urban3

Para esta secuencia la aptitud del mejor individuo obtenida fue de 4.8766. El error angular promedio en la estimación del flujo para esta secuencia fué de 31.2206, mientras que el error de punto final de 5.0296.

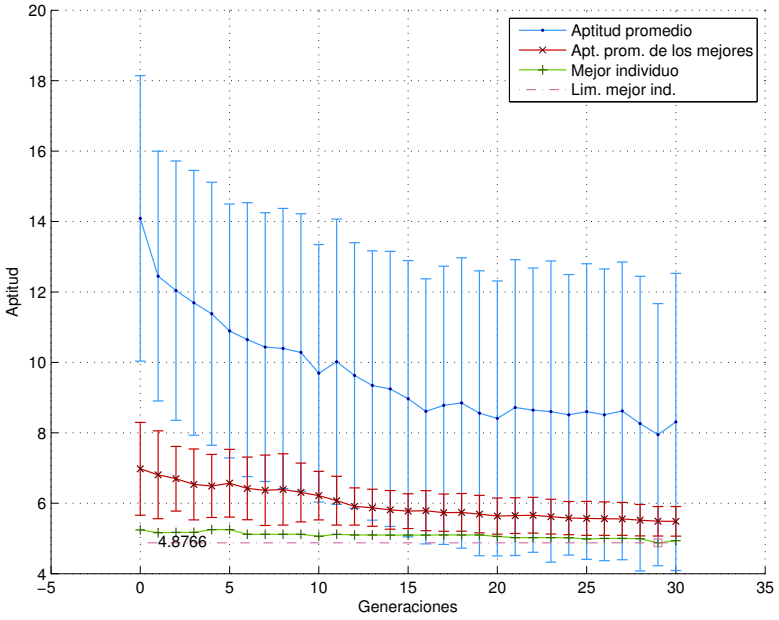


Figura 46. Evolución promedio de 30 ejecuciones del programa de GP para la secuencia Urban3

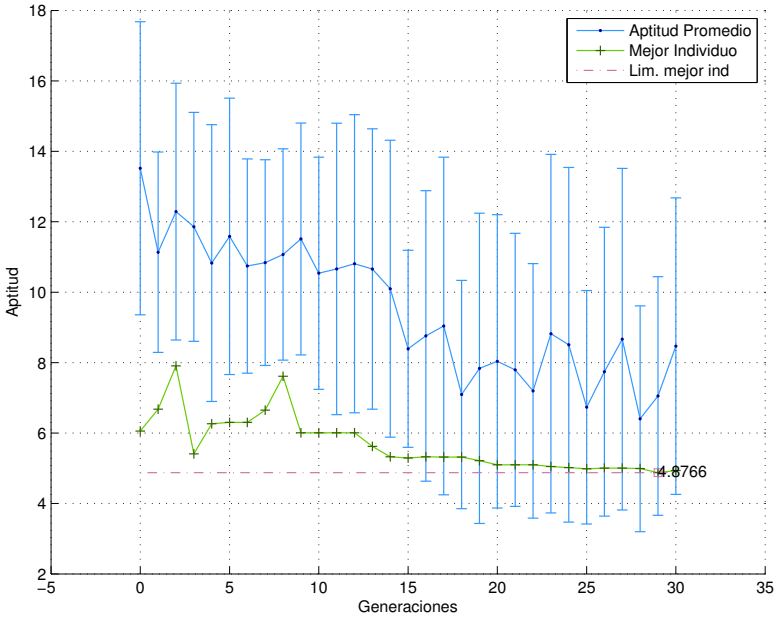


Figura 47. Evolución de la mejor ejecución para la secuencia Urban3

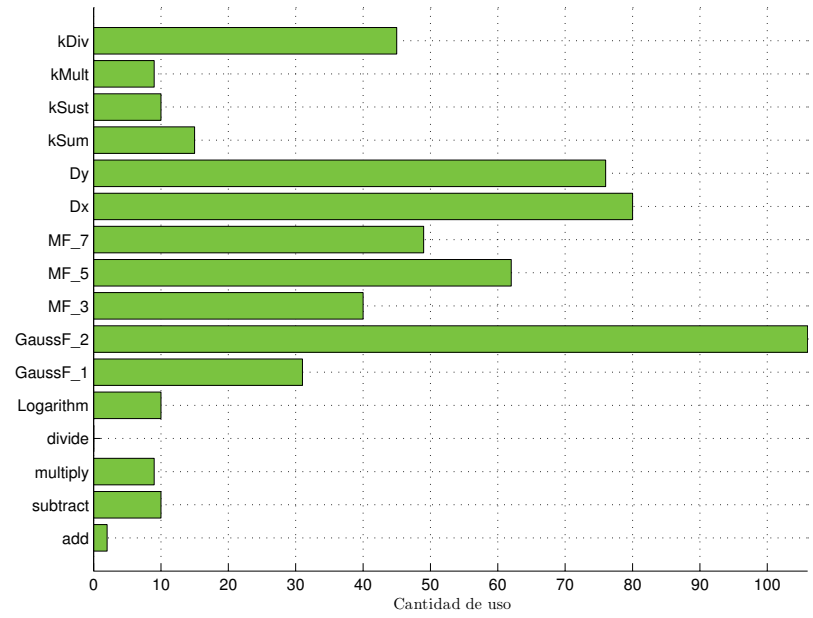


Figura 48. Frecuencia de uso de funciones para la secuencia Urban2

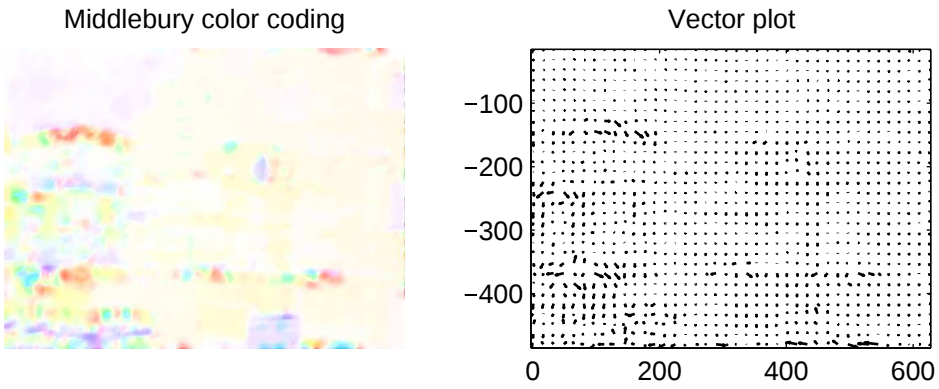


Figura 49. Flujo estimado para la secuencia Urban3 — AAE 31.2206 EPE 5.0296

5.1.1.8. Resultados de la secuencia Venus

Para esta secuencia la aptitud del mejor individuo obtenida fue de 3.4114. El error angular promedio en la estimación del flujo para esta secuencia fué de 19.0584, mientras que el error de punto final de 1.4211.

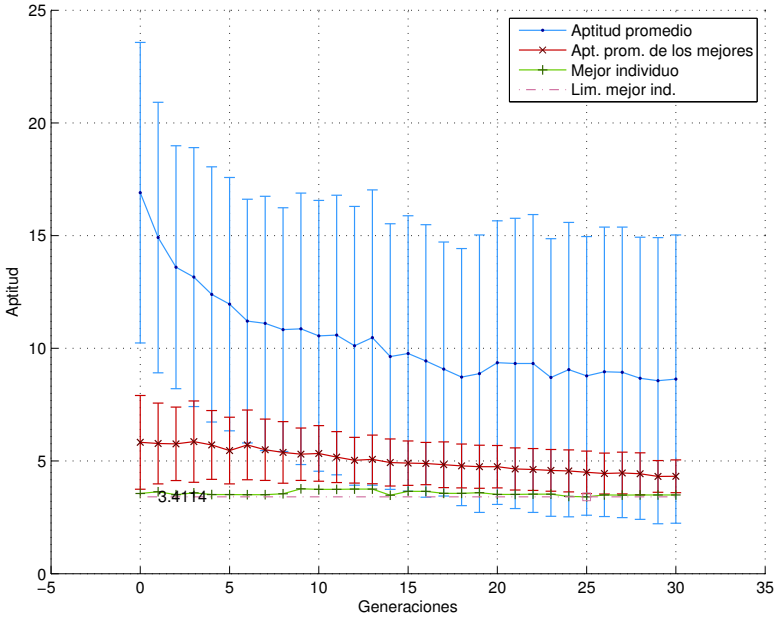


Figura 50. Evolución promedio de 30 ejecuciones del programa de GP para la secuencia Venus

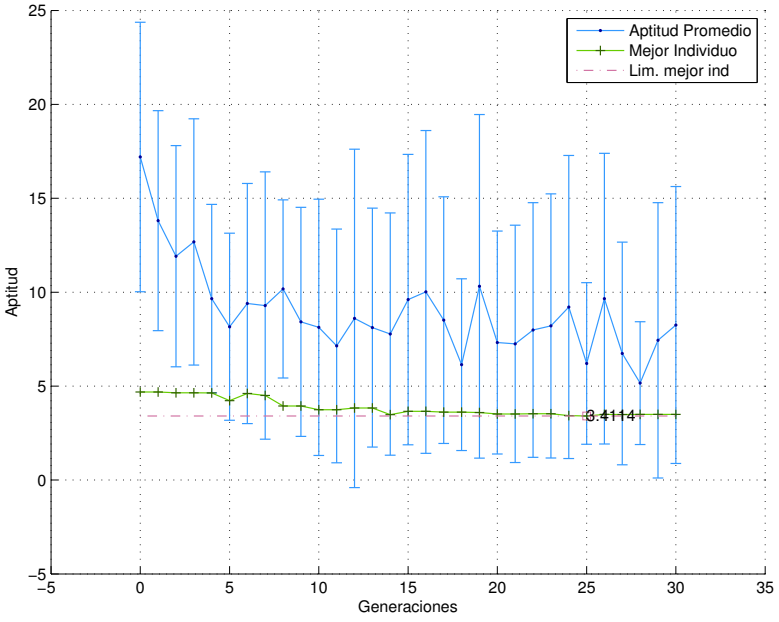


Figura 51. Evolución de la mejor ejecución para la secuencia Venus

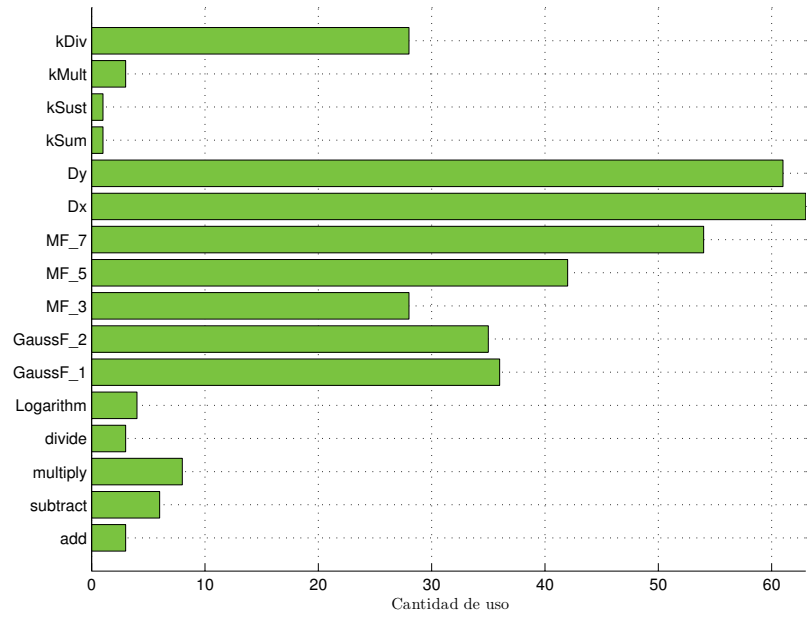


Figura 52. Frecuencia de uso de funciones para la secuencia Venus

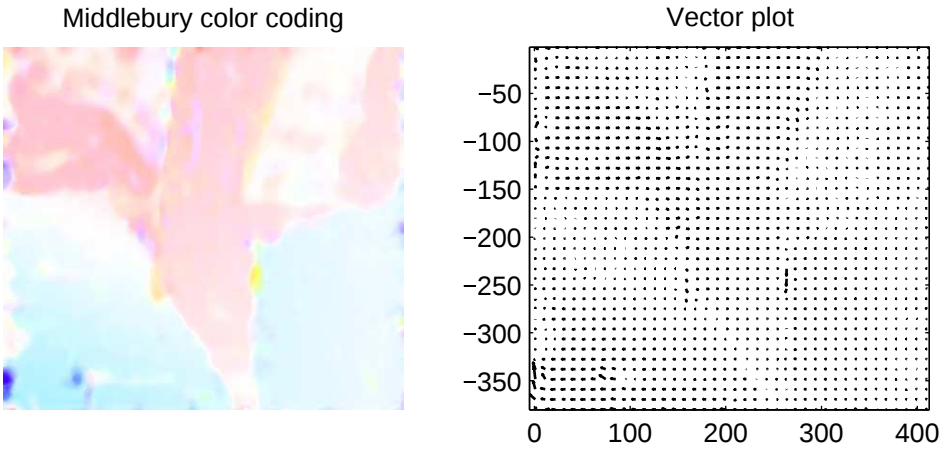


Figura 53. Flujo estimado para la secuencia Venus — AAE 19.0584 EPE 1.4211

5.1.2. gpPLKColor

A continuación se muestran los resultados del programa de GP en su segunda versión, la cual integra información de las secuencias de imágenes en distintos espacios de color para la estimación del flujo óptico. Los parámetros empleados para el programa de GP se muestran en la siguiente tabla.

Tabla 10. Parametros gpPLKColor

Número de ejecuciones	15	
Tamaño de la población	30	
Método de inicialización	Ramped Half-and-Half	
Número de generaciónes	30	
Número máximo de árboles por individuo	6	
Profundidad de los árboles	5	
Profundidad máxima dinámica	7	
Profundidad máxima real	9	
Probabilidades de operadores genéticos	Cruce nivel gen. = 0.8	Mutanción nivel gen. = 0.8
	Mut. nivel gen. = 0.2	Mut. nivel gen. = 0.2
Selección	Torneo Lexicográfico	
Supervivencia	Reemplazo generacional	

5.1.2.1. Resultados de la secuencia Dimetrodon

La gráfica de la Figura 54 muestra la estadística de la evolución de 15 ejecuciones del programa de GP evaluando la secuencia Dimetrondon. Para esta secuencia la aptitud del mejor individuo obtenida fue de 1.5207. La aptitud máxima que puede obtener un individuo al ser evaluado para todos los casos es 0. La Figura55 corresponde a la frecuencia de uso de funciones y terminales del operador de color, mientras que la Figura 56 muestra el uso de funciones del operador para la correción del flujo, debido a que el conjunto de terminales descrito en la Tabla 3 resulta de aplicar primitivas del conjunto de funciones, por lo tanto la gráfica para este conjunto se omite.

La Figura 57 representa el uso de funciones y terminales para la función de integracion, que establece la manera en la que se combinan los operadores para la correción de flujo. Finalmente, en la Figura 58 se muestra el resultado del flujo óptico estimado para esta secuencia, en este caso se obtuvo un error angular promedio en la estimación de 3.4191 y un error de punto final

de 0.1686. Las gráficas correspondientes a la evaluación de las secuencias restantes siguen el mismo orden.

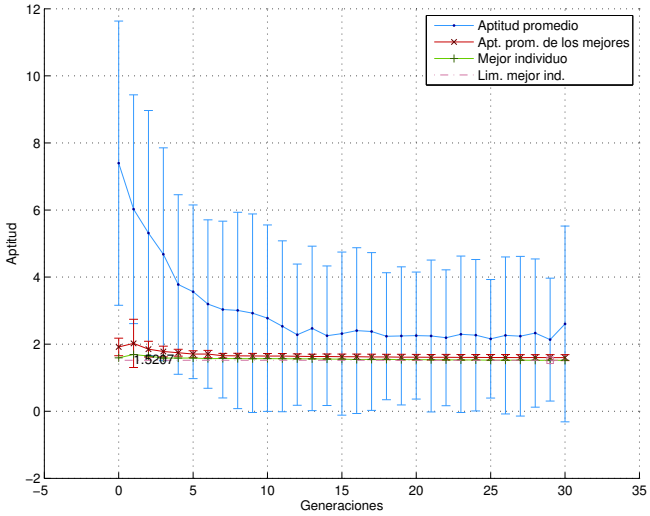
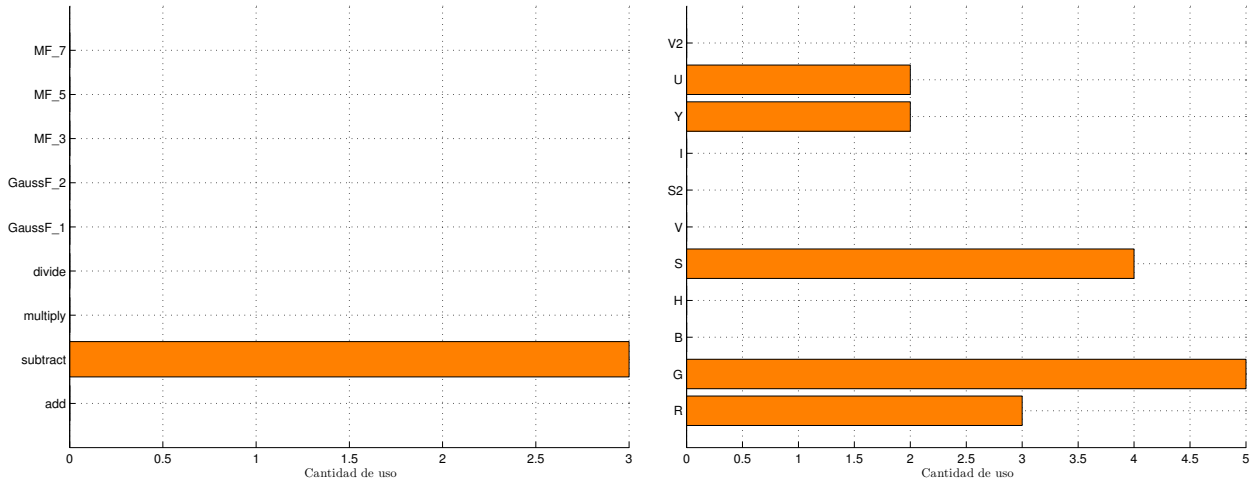


Figura 54. Evolución promedio de 15 ejecuciones del programa de GP para la secuencia Dimetrodon



(a) Funciones **(b)** Terminales

Figura 55. Frecuencia de uso de funciones y terminales del operador de color en la secuencia Dimetrodon

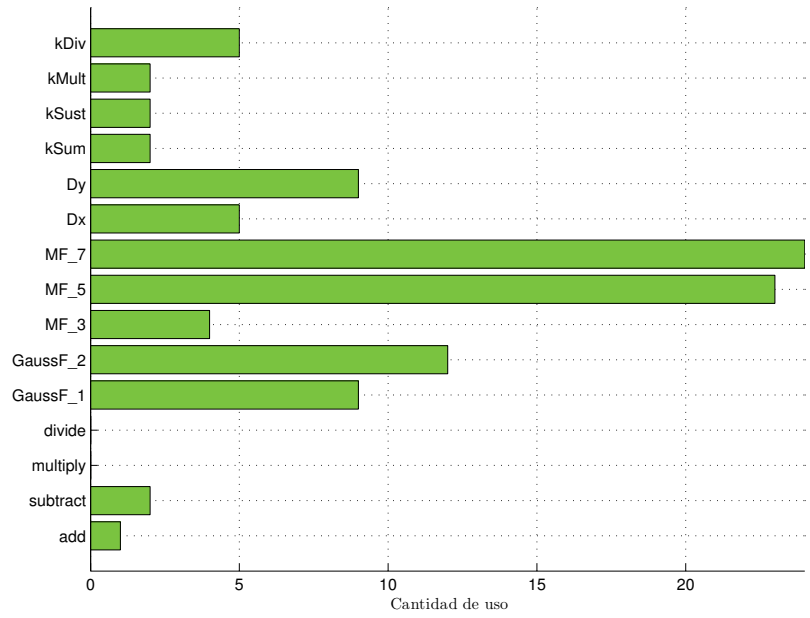
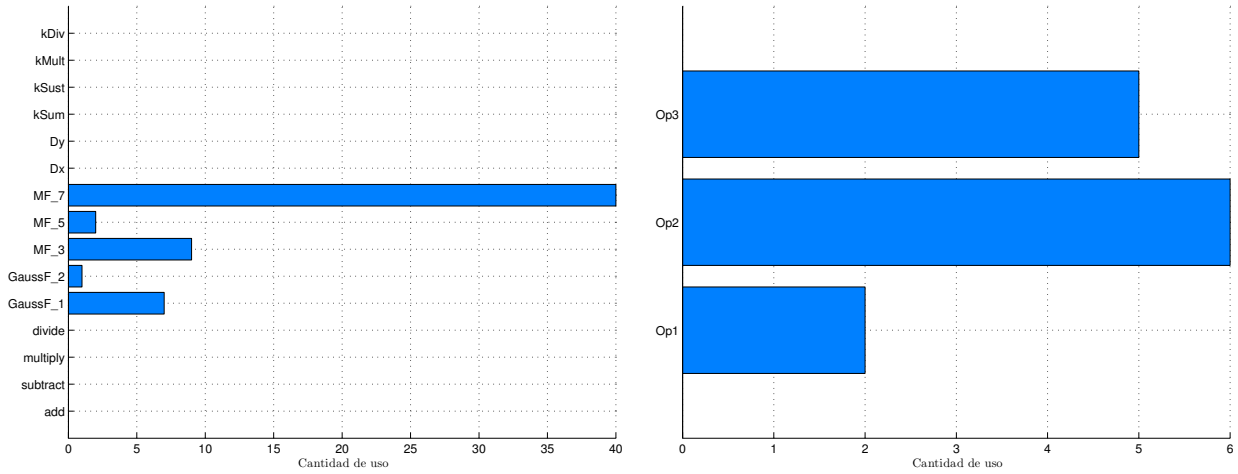


Figura 56. Cantidad de uso de funciones del operador de flujo en la secuencia Dimetrodon



(a) Funciones **(b)** Terminales

Figura 57. Frecuencia de uso de funciones y terminales de la función de integración en la secuencia Dimetrodon

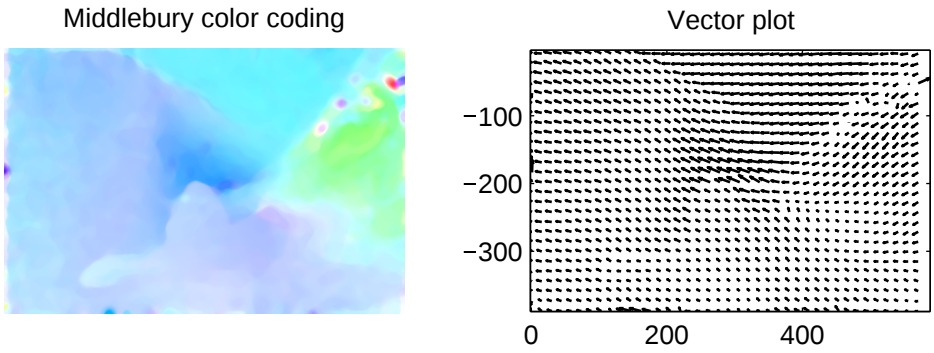


Figura 58. Flujo estimado para la secuencia Dimetrodon— AAE 3.4191 EPE 0.1686

5.1.2.2. Resultados para la secuencia Groove2

Para esta secuencia la aptitud del mejor individuo obtenida fue de 4.8433. El error angular promedio en la estimación del flujo para esta secuencia fué de 4.4269, mientras que el error de punto final de 0.3033.

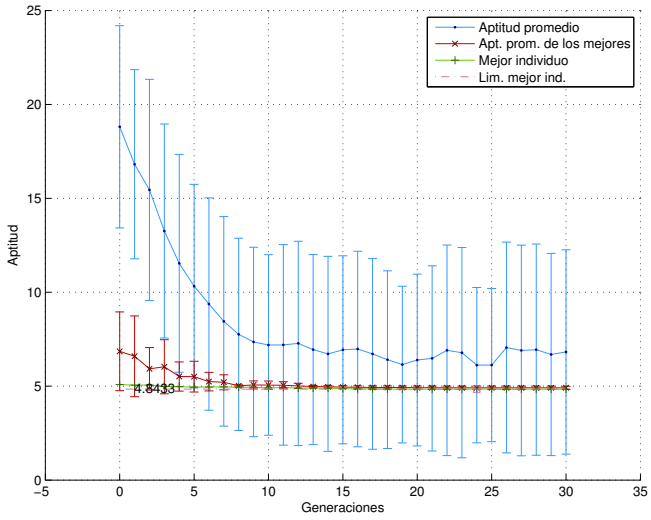
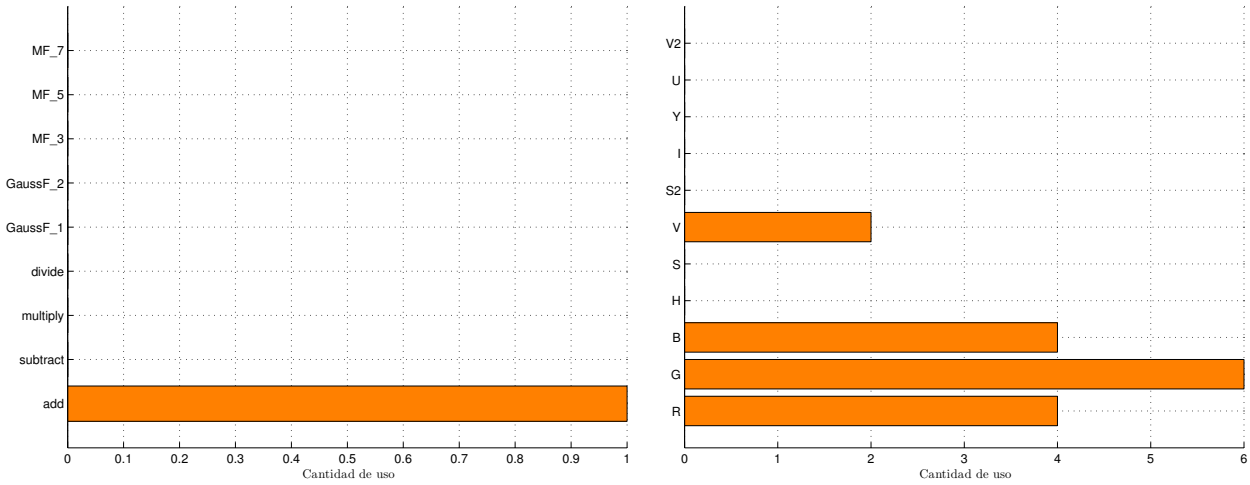


Figura 59. Evolución promedio de 15 ejecuciones del programa de GP para la secuencia Groove2



(a) Funciones **(b)** Terminales

Figura 60. Frecuencia de uso de funciones y terminales del operador de color en la secuencia Groove2

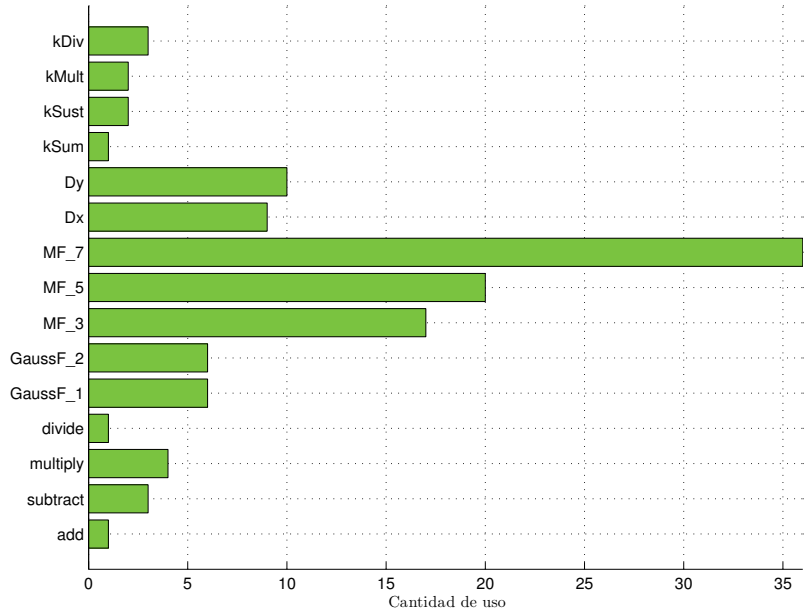
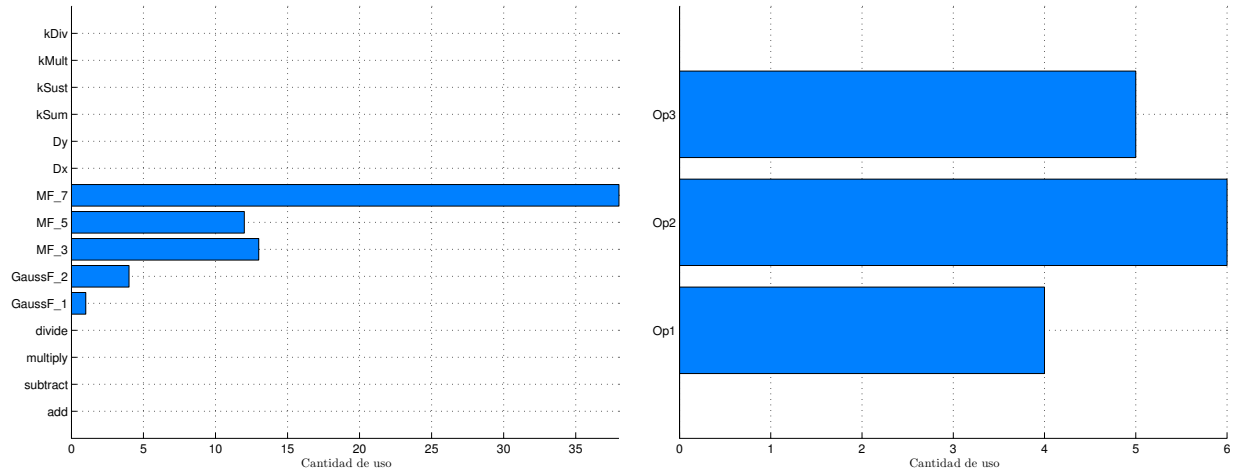


Figura 61. Frecuencia de uso de funciones del operador de flujo en la secuencia Groove2



(a) Funciones **(b)** Terminales

Figura 62. Frecuencia de uso de funciones y terminales de la función de integración en la secuencia Groove2

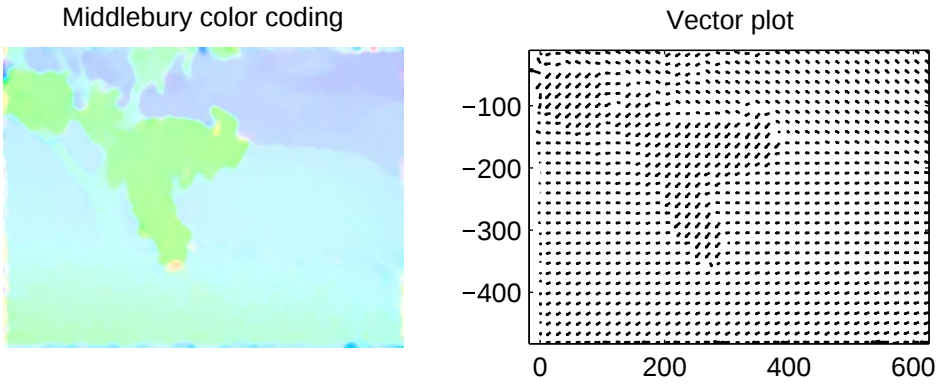


Figura 63. Flujo estimado para la secuencia Dimetrodon— AAE 4.4269 EPE 0.30338

5.1.2.3. Resultados para la secuencia Groove3

Para esta secuencia la aptitud del mejor individuo obtenida fue de 8.0578. El error angular promedio en la estimación del flujo para esta secuencia fué de 11.579, mientras que el error de punto final de 1.2465.

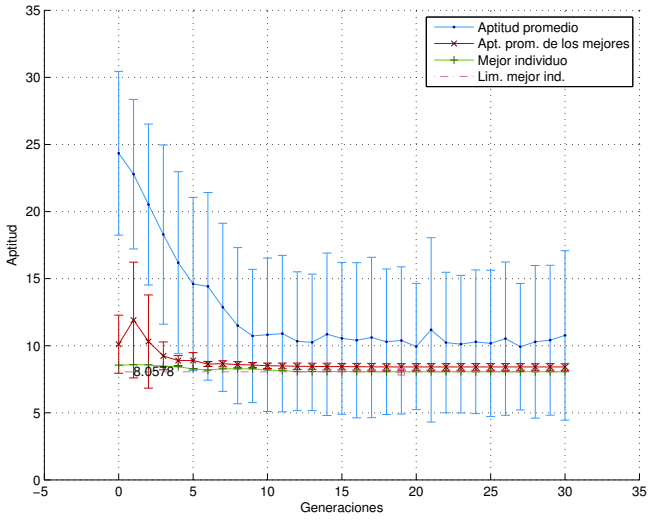
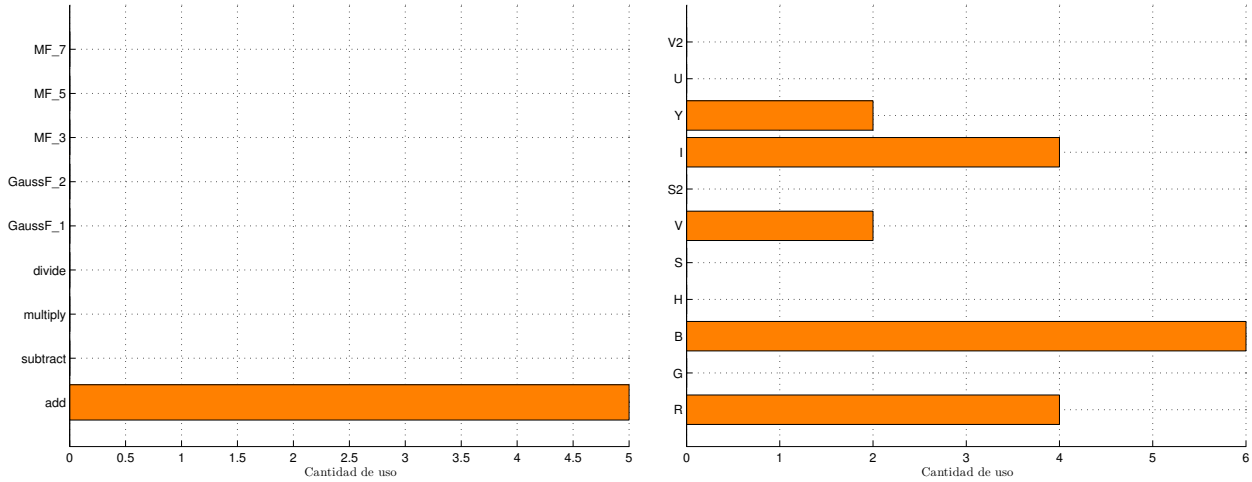


Figura 64. Evolución promedio de 15 ejecuciones del programa de GP para la secuencia Groove3



(a) Funciones **(b)** Terminales

Figura 65. Frecuencia de uso de funciones y terminales del operador de color en la secuencia Groove3

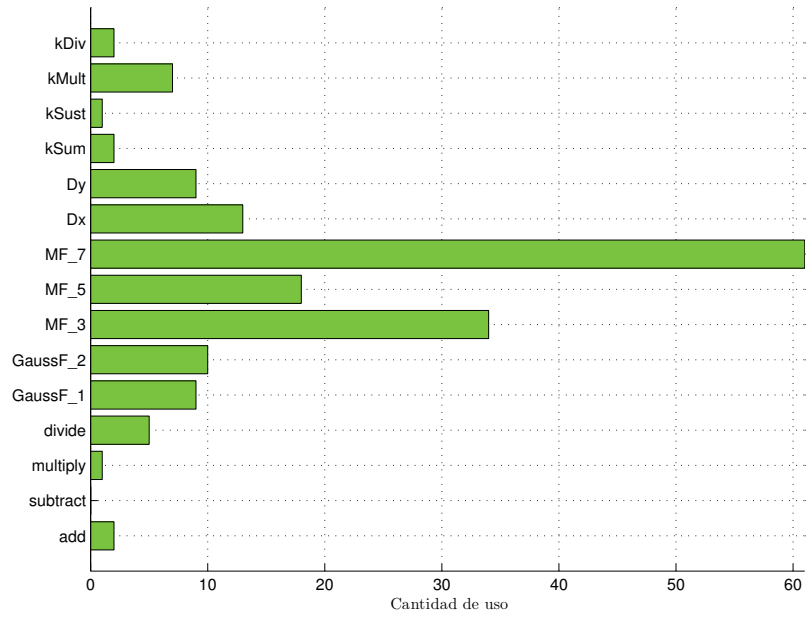
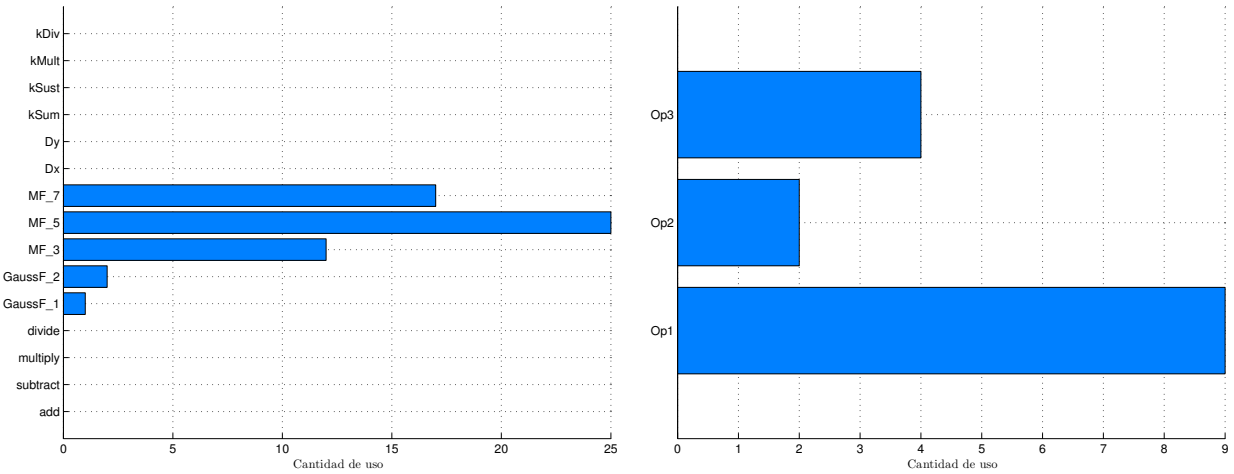


Figura 66. Frecuencia de uso de funciones del operador de flujo en la secuencia Groove3



(a) Funciones **(b)** Terminales

Figura 67. Frecuencia de uso de funciones y terminales de la función de integración en la secuencia Groove3

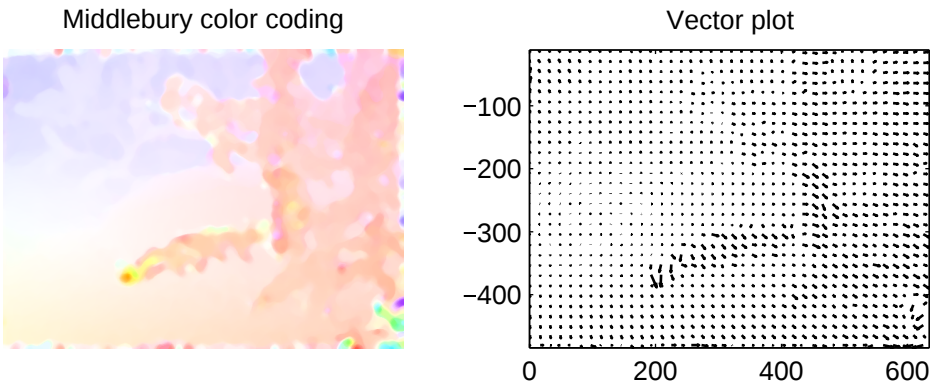


Figura 68. Flujo estimado para la secuencia Groove3— AAE 11.579 EPE 1.2465

5.1.2.4. Resultados para la secuencia Hydrangea

Para esta secuencia la aptitud del mejor individuo obtenida fue de 3.2243. El error angular promedio en la estimación del flujo para esta secuencia fué de 5.1464, mientras que el error de punto final de 0.4610.

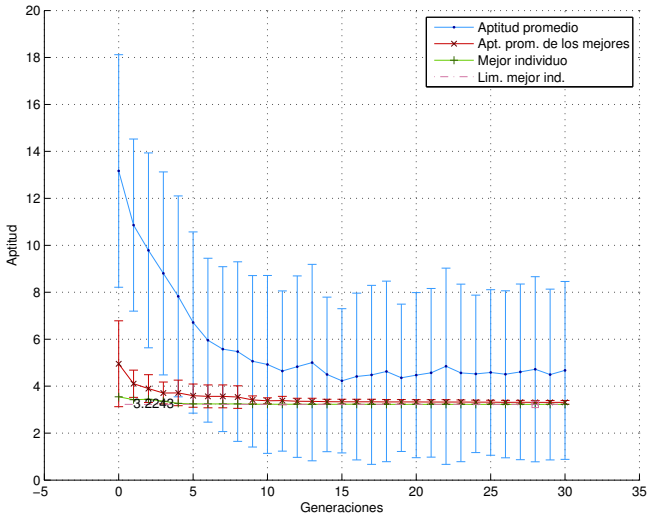
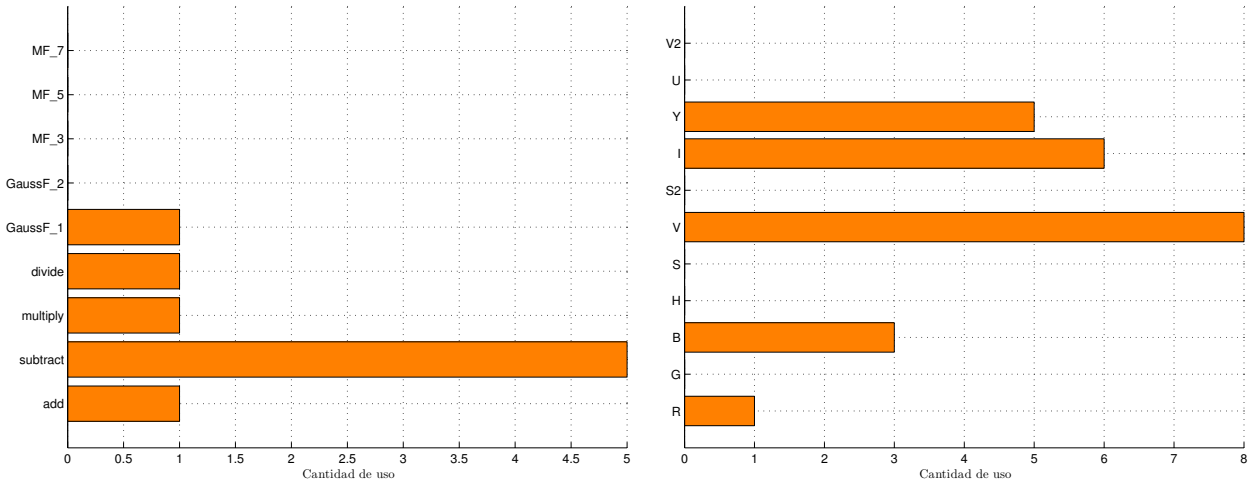


Figura 69. Evolución promedio de 15 ejecuciones del programa de GP para la secuencia Hydrangea



(a) Funciones **(b)** Terminales

Figura 70. Frecuencia de uso de funciones y terminales del operador de color en la secuencia Hydrangea

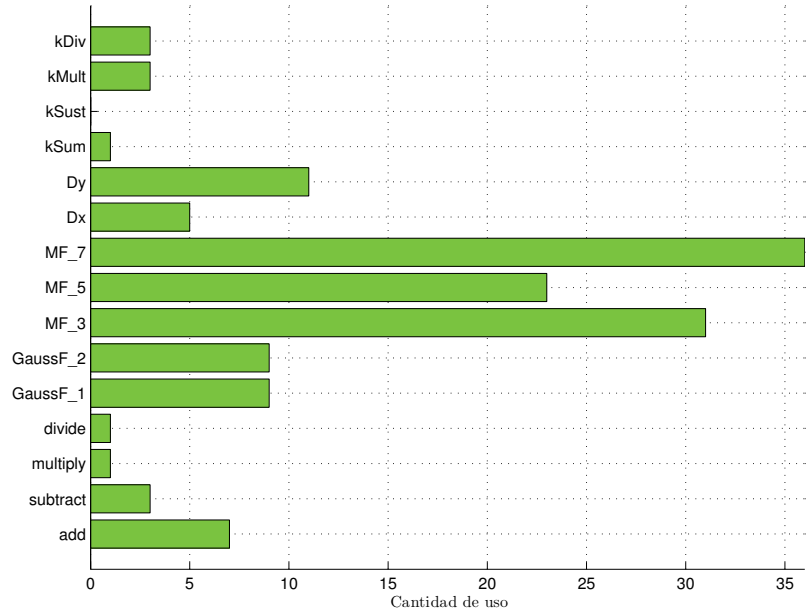
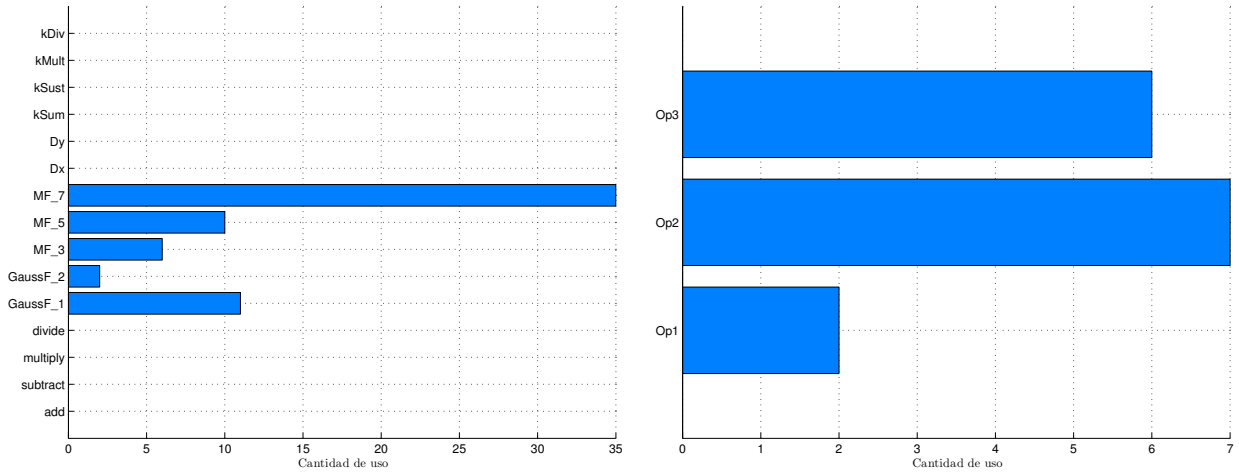


Figura 71. Frecuencia de uso de funciones del operador de flujo en la secuencia Hydrangea



(a) Funciones **(b)** Terminales

Figura 72. Frecuencia de uso de funciones y terminales de la función de integración en la secuencia Hydrangea

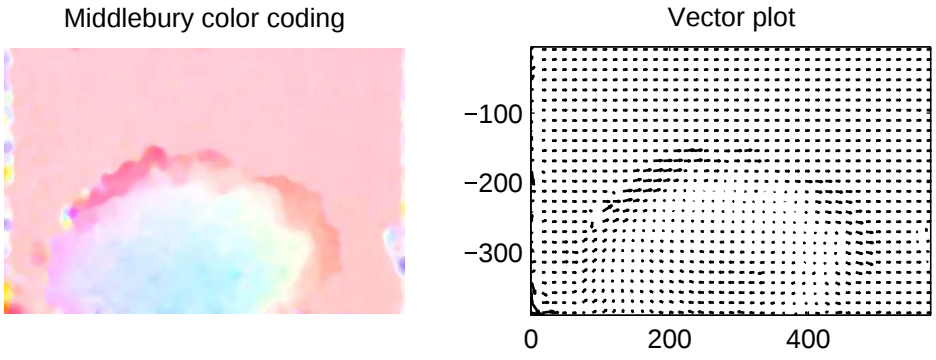


Figura 73. Flujo estimado para la secuencia Hydrangea— AAE 5.1464 EPE 0.46108

5.1.2.5. Resultados para la secuencia RubberWhale

Para esta secuencia la aptitud del mejor individuo obtenida fue de 1.3953. El error angular promedio en la estimación del flujo para esta secuencia fué de 8.2622, mientras que el error de punto final de 0.2841.

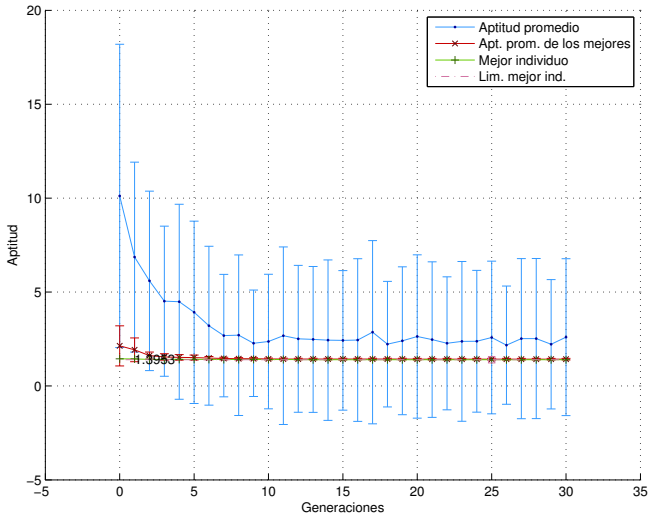
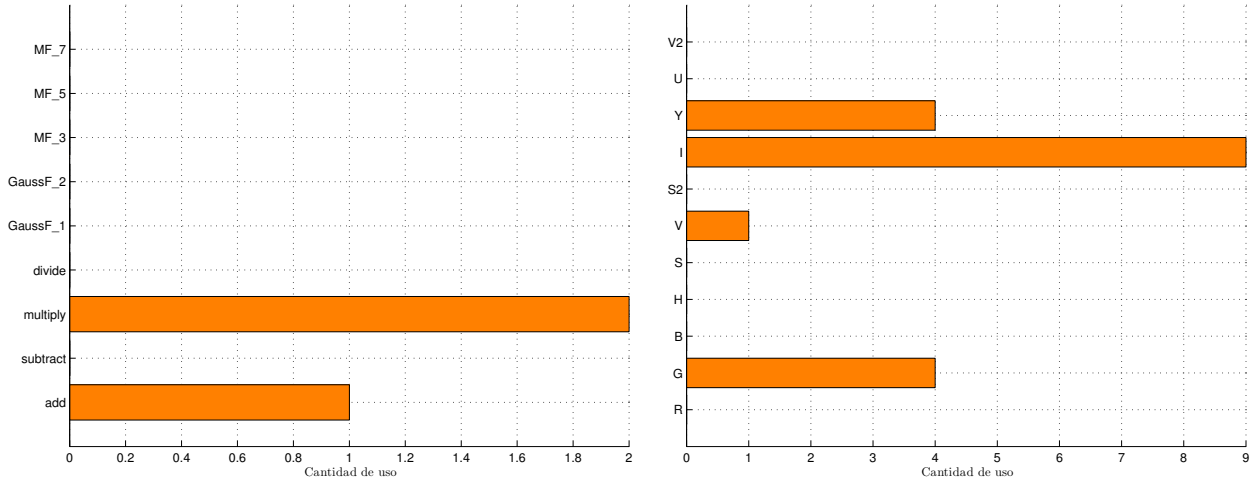


Figura 74. Evolución promedio de 15 ejecuciones del programa de GP para la secuencia RubberWhale



(a) Funciones **(b)** Terminales

Figura 75. Frecuencia de uso de funciones y terminales del operador de color en la secuencia RubberWhale

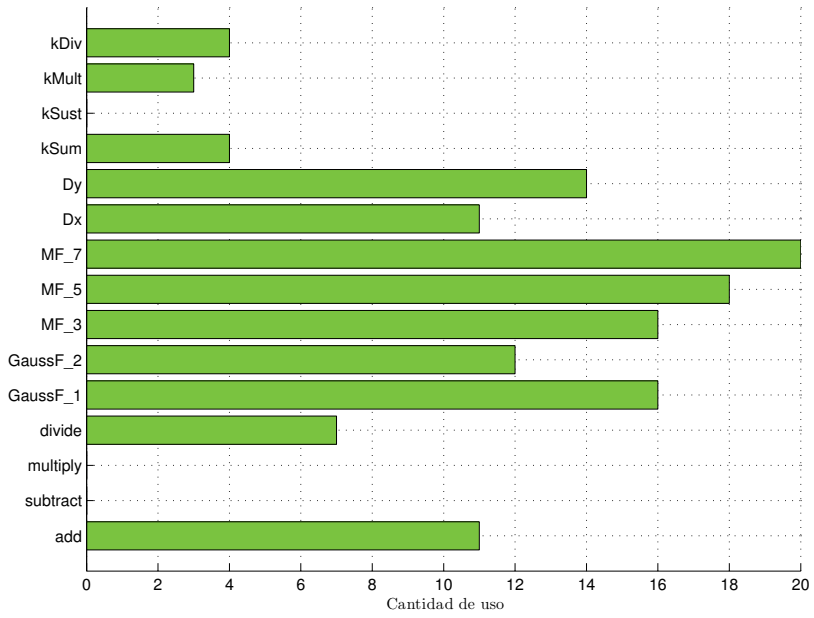
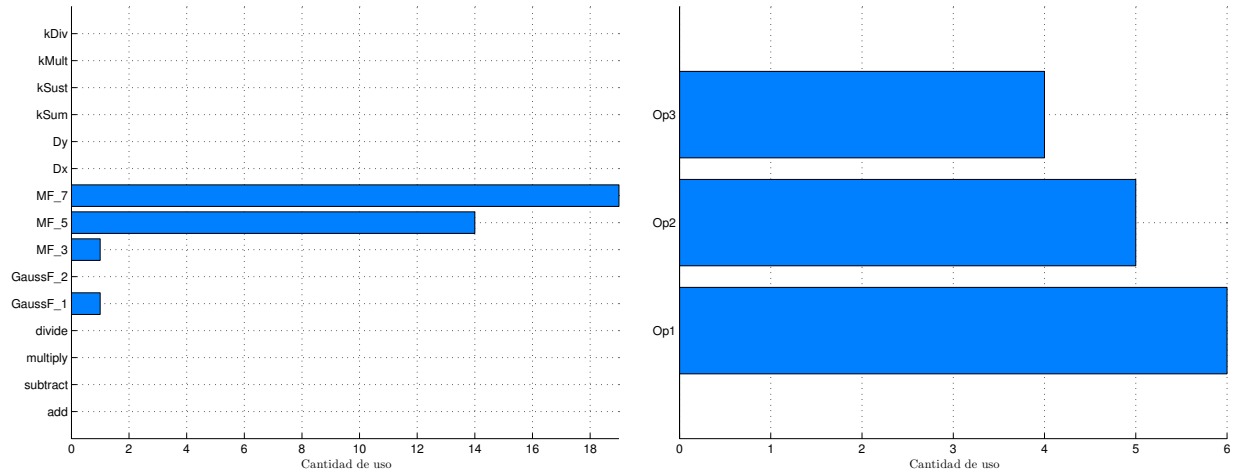


Figura 76. Frecuencia de uso de funciones del operador de flujo en la secuencia RubberWhale



(a) Funciones **(b)** Terminales

Figura 77. Frecuencia de uso de funciones y terminales de la función de integración en la secuencia RubberWhale

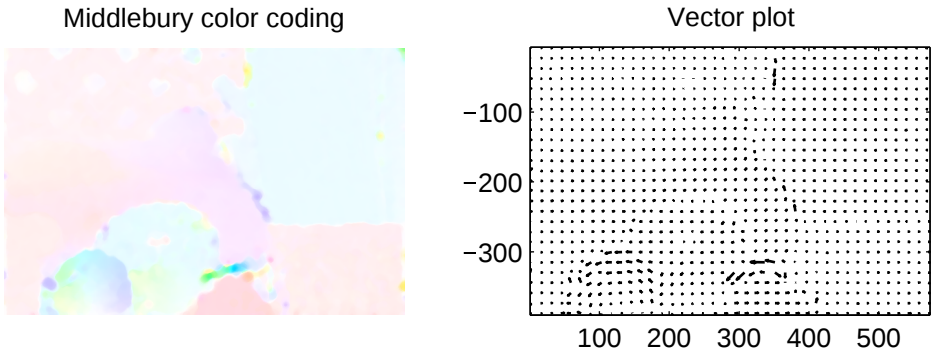


Figura 78. Flujo estimado para la secuencia RubberWhale— AAE 8.2622 EPE 0.2841

5.1.2.6. Resultados para la secuencia Urban2

Para esta secuencia la aptitud del mejor individuo obtenida fue de 5.7834. El error angular promedio en la estimación del flujo para esta secuencia fué de 23.5533, mientras que el error de punto final de 5.407.

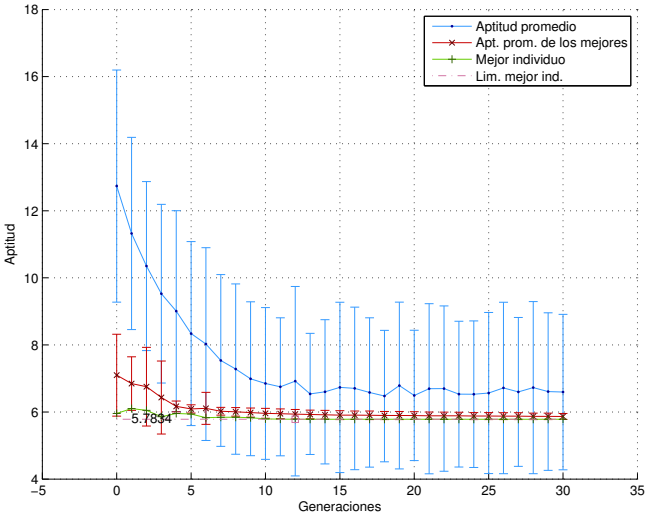
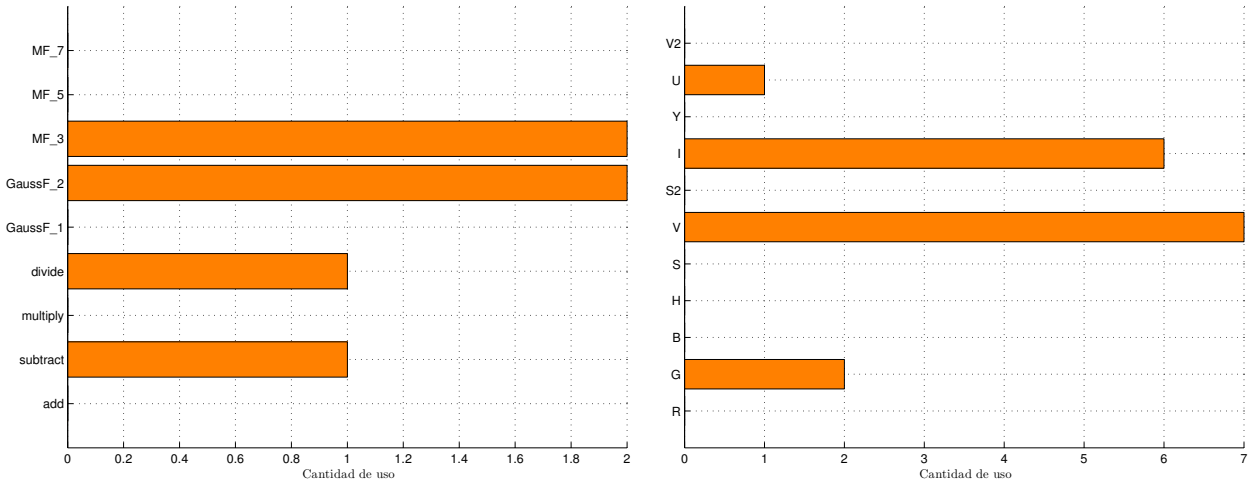


Figura 79. Evolución promedio de 15 ejecuciones del programa de GP para la secuencia Urban2



(a) Funciones **(b)** Terminales

Figura 80. Frecuencia de uso de funciones y terminales del operador de color en la secuencia Urban2

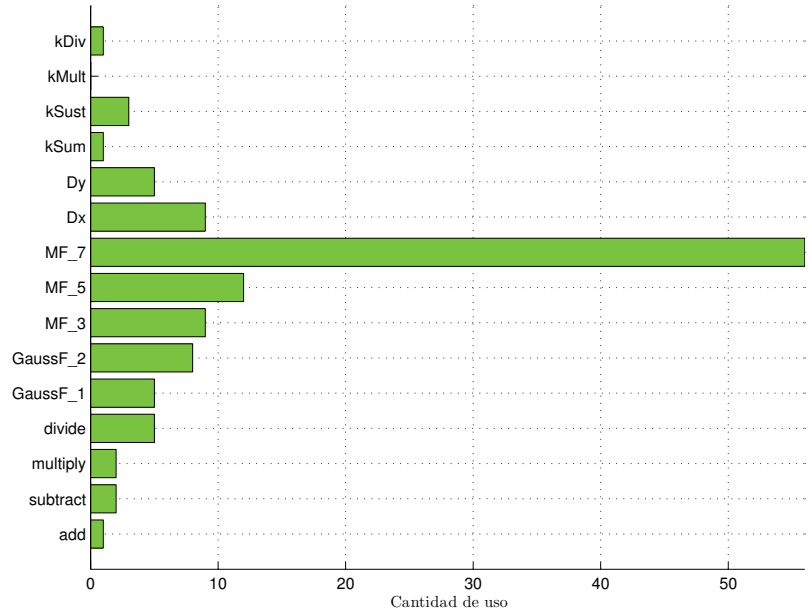
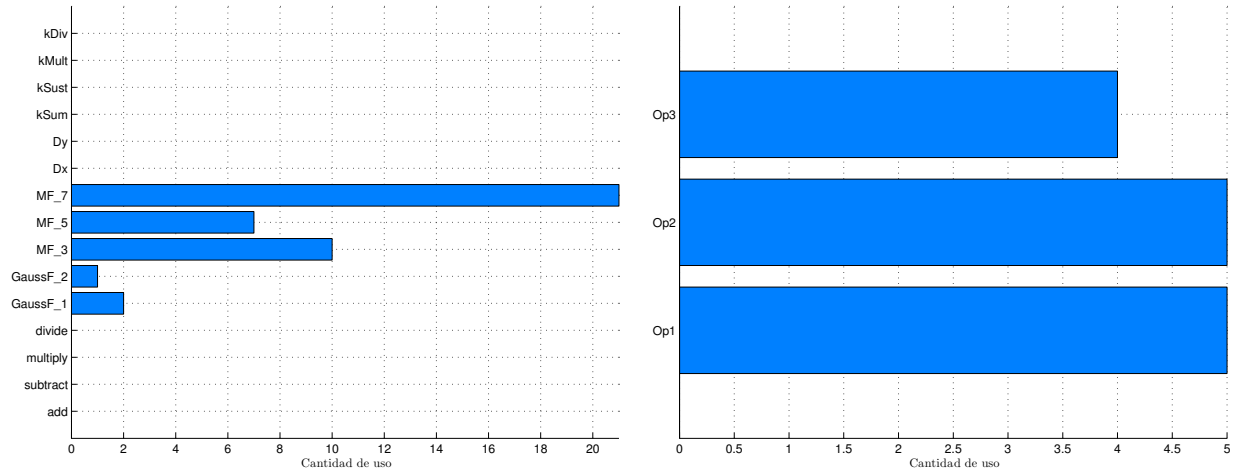


Figura 81. Frecuencia de uso de funciones del operador de flujo en la secuencia Urban2



(a) Funciones **(b)** Terminales

Figura 82. Frecuencia de uso de funciones y terminales de la función de integración en la secuencia Urban2

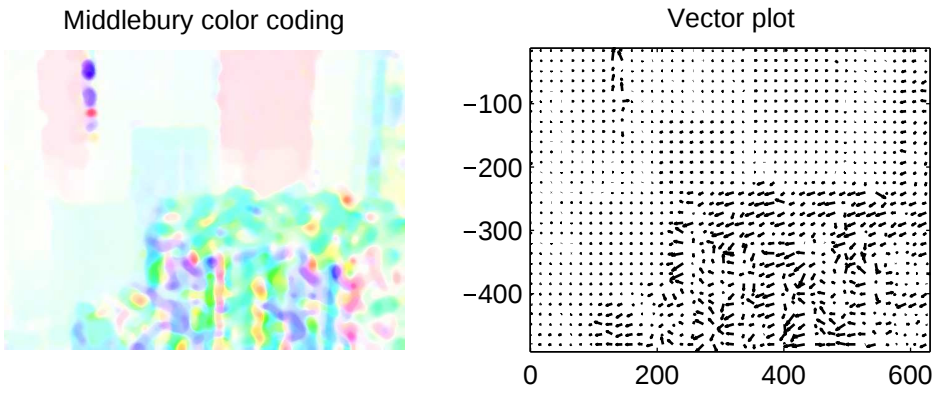


Figura 83. Flujo estimado para la secuencia Urban2— AAE 23.5533 EPE 5.407

5.1.2.7. Resultados para la secuencia Urban3

Para esta secuencia la aptitud del mejor individuo obtenida fue de 4.6339. El error angular promedio en la estimación del flujo para esta secuencia fué de 26.1154, mientras que el error de punto final de 5.0262.

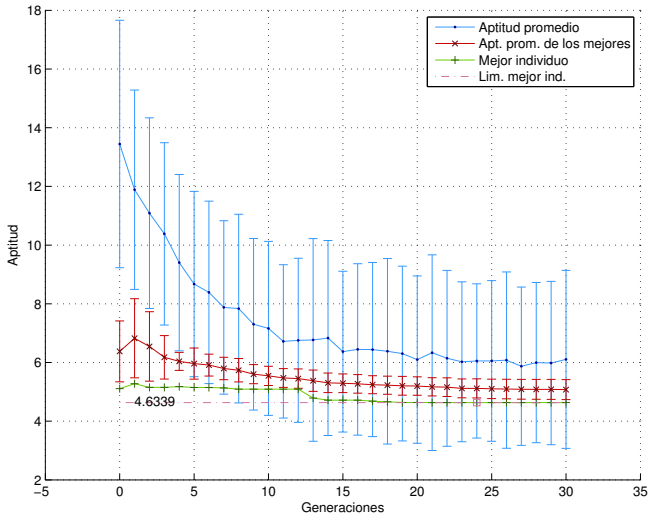
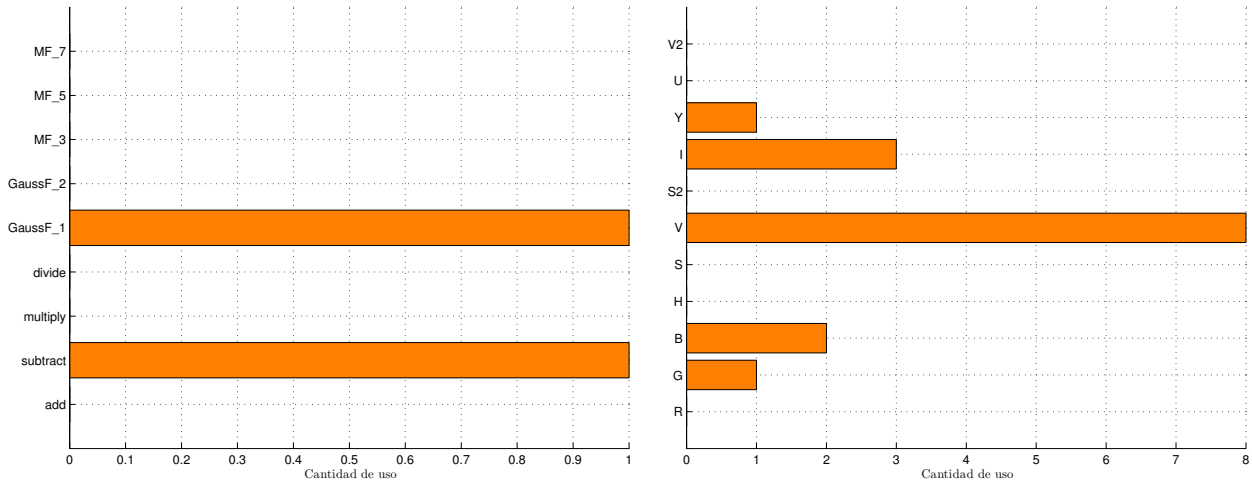


Figura 84. Evolución promedio de 15 ejecuciones del programa de GP para la secuencia Urban3



(a) Funciones **(b)** Terminales

Figura 85. Frecuencia de uso de funciones y terminales del operador de color en la secuencia Urban3

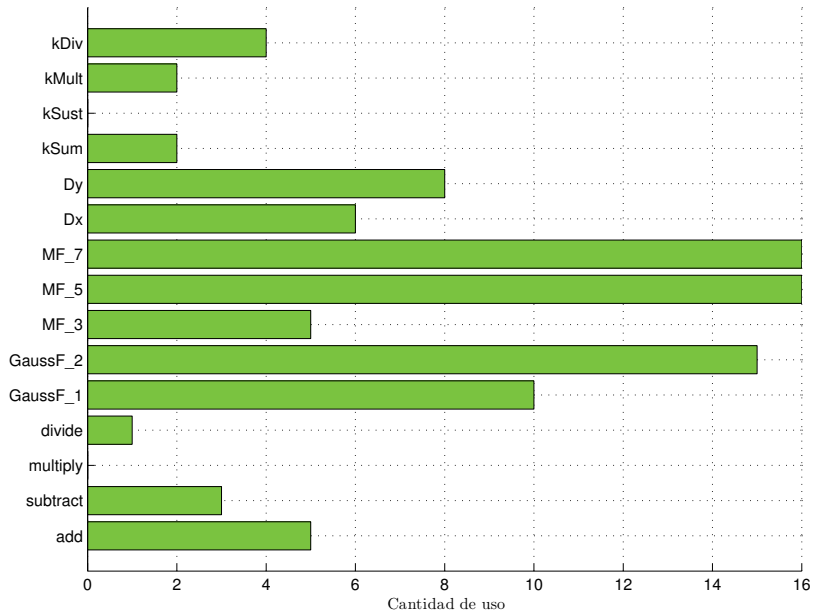
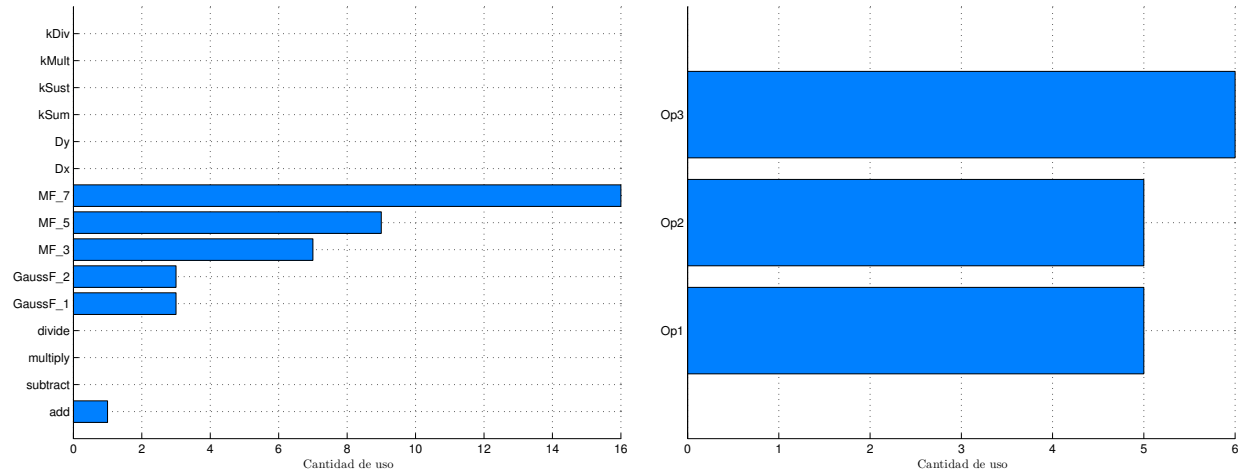


Figura 86. Frecuencia de uso de funciones del operador de flujo en la secuencia Urban3



(a) Funciones **(b)** Terminales

Figura 87. Frecuencia de uso de funciones y terminales de la función de integración en la secuencia Urban3

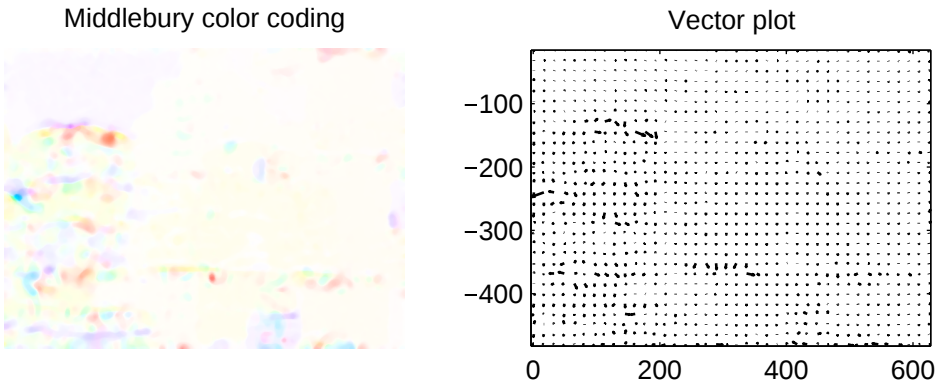


Figura 88. Flujo estimado para la secuencia Urban3— AAE 26.1154 EPE 5.0262

5.1.2.8. Resultados para la secuencia Venus

Para esta secuencia la aptitud del mejor individuo obtenida fue de 3.2367. El error angular promedio en la estimación del flujo para esta secuencia fué de 12.4474, mientras que el error de punto final de 0.9174.

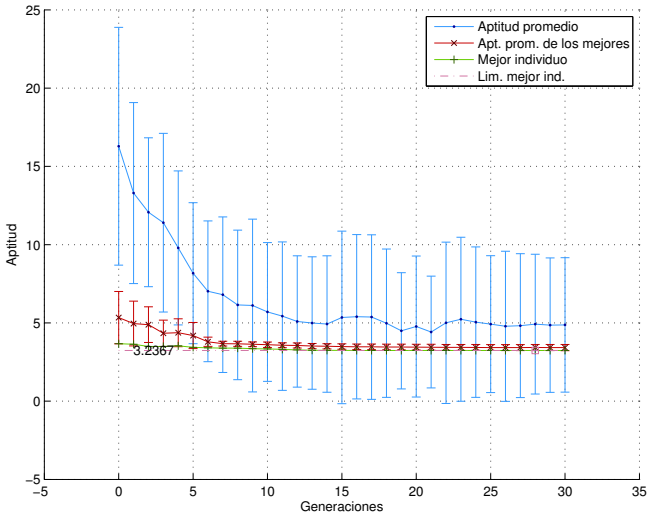
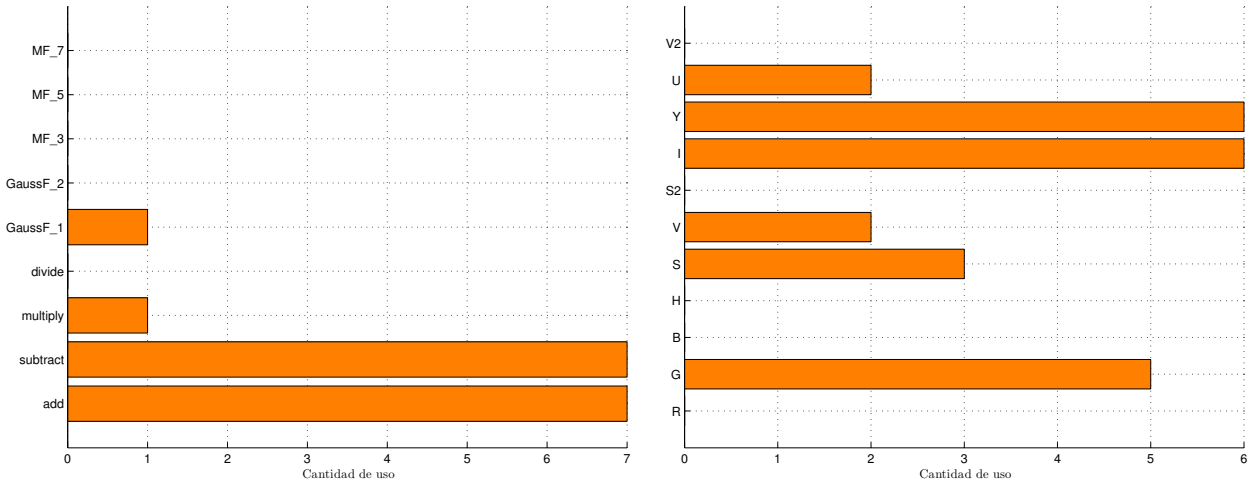


Figura 89. Evolución promedio de 15 ejecuciones del programa de GP para la secuencia Venus



(a) Funciones **(b)** Terminales

Figura 90. Frecuencia de uso de funciones y terminales del operador de color en la secuencia Venus

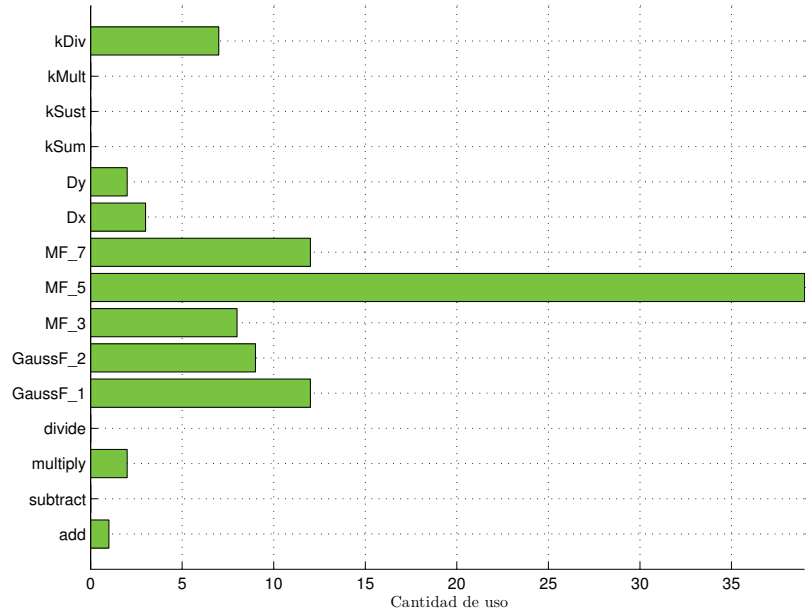
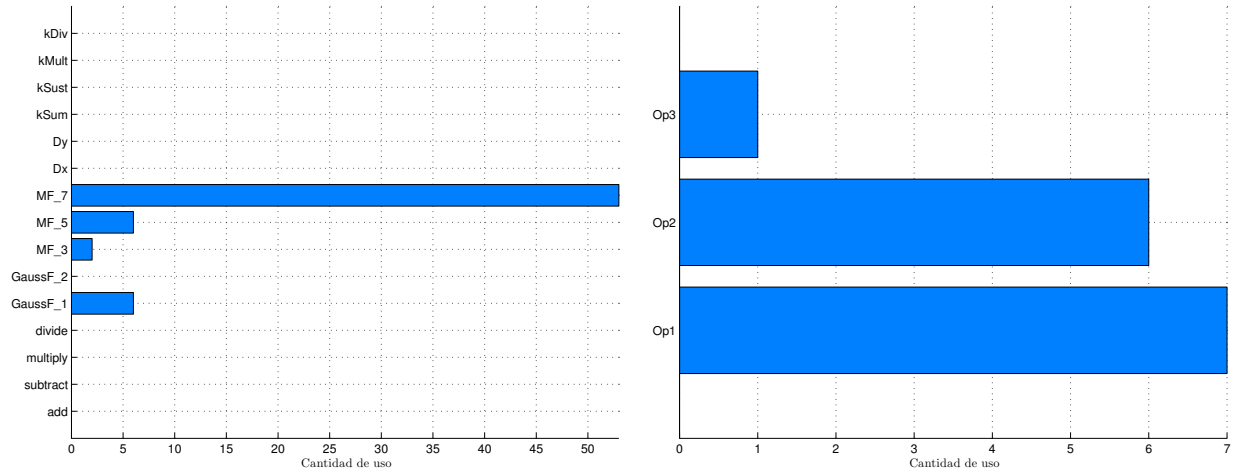


Figura 91. Frecuencia de uso de funciones del operador de flujo en la secuencia Venus



(a) Funciones **(b)** Terminales

Figura 92. Frecuencia de uso de funciones y terminales de la función de integración en la secuencia Venus

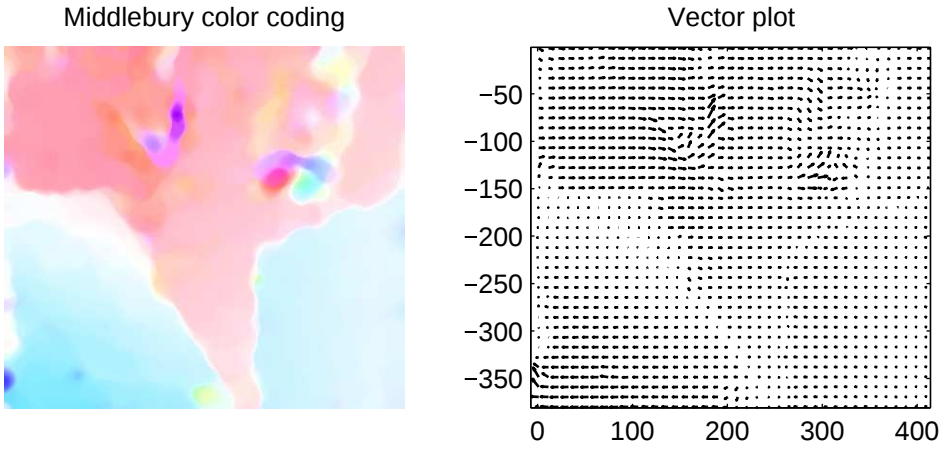


Figura 93. Flujo estimado para la secuencia Venus— AAE 12.4474 EPE 0.9174

Capítulo 6. Conclusiones

En este capítulo se presentan las concluciones y algunas propuestas como trabajo futuro.

6.1. Conclusiones

En este trabajo de tesis se abordó el problema de la estimación del flujo óptico, que es uno de los problemas en visión por computadora que en las últimas décadas ha tenido un avance importante en cuanto a métodos para poder estimarlo cada vez con mayor precisión. Se propuso un método que mediante programación genética genera operadores de manera automática que actuan sobre el flujo óptico de una secuencia de imágenes, mejorando asi la precisión de la estimación.

Como parte de los primeros experimentos realizados usando los métodos de HS y LK, se concluyó que el enfoque multi-resolución proporciona una estimación del flujo óptico con un error menor comparado con la formulación tradicional. Es por esto que se eligió este enfoque junto con el método de LK, sumado a la simplicidad para su implementación.

Del método propuesto se derivaron dos variantes, una de ellas genera operadores de manera automática para aplicarlos al flujo óptico para secuencias en escala de grises. Después de haber realizados los experimentos con la primer variante del método propuesto evaluando el conjunto de secuencias de entrenamiento de la base de datos de Middlebury para flujo óptico; se decidió incorpora secuencias de imágenes en diferentes espacios de color (*RGB, HSV,HSI,YUV*) para la segunda versión, con la finalidad de agregar diversidad en las posibles soluciones generadas.

Comparando los resultados obtenidos, la segunda versión propuesta que es la que integra las secuencias de imagenes con información a color ofrece una mejor estimación del flujo óptico. Debido a la naturaleza del algoritmo de GP para cada secuencia evaluada se obtuvo un individuo diferente(que representa un conjunto de operadores) por lo que se realizaron algunos experimentos más; esto con la finalidad de saber si existía uno o varios individuos que brindaran una estimación mejor en el flujo óptico para todas las secuencias. Después de realizar los experimentos se puede concluir que si es posible esto para unos casos, lo cúal resulta un tanto interesante ya que un operador que no surgió del proceso evolutivo para una secuencia dada, puede ofrecer un mejor rendimiento en comparación con otro que si fue evolucionado para esa secuencia en específico.

Del conjunto de 8 secuencias evaluadas compuesto por imágenes naturales y sintéticas, la mejora considerable en la estimación del flujo óptico se obtuvo para la secuencia Dimetrodon con un error angular promedio de 2.9467 y un error de punto final de 0.1517. Si bien la precisión en la estimación para el resto de las secuencias es menor a comparación del método usado como referencia, los resultados obtenidos mediante el método propuesto es mejor a los que arroja el método de LK en su versión multi-resolución que se utilizó como método base en este trabajo.

6.2. Trabajo Futuro

Considerando los resultados obtenidos en este trabajo de tesis, se pueden hacer algunos ajustes con la finalidad de mejorarlos.

Una de las propuestas es realizar nuevamente la ejecución del programa de GP considerando únicamente las funciones y terminales de mayor uso en la construcción de los operadores, basado en las frecuencias de uso obtenidas con los experimentos realizados. Lo anterior con la finalidad de tratar de disminuir el tiempo de ejecución para el programa de GP, que en promedio fué de nueve horas para la versión del método propuesto que integra secuencias de imágenes a color.

Considerando el tiempo de computo mencionado anteriormente, se propone realizar la implementación del programa de GP utilizando técnicas de computo paralelo, como podría ser el uso de las GPUs para la parte de evaluación de la población. También podrían utilizarse técnicas de computo distribuido, para realizar la ejecución del programa de GP en más de una estación de trabajo a la vez.

Por último, se propone realizar la implementación de este método utilizando un enfoque multi-objetivo, con el propósito de encontrar individuos que mejoren los resultados reportados en este trabajo de tesis al disminuir el error angular promedio y el error de punto final en la estimación del flujo óptico. Lo anterior no se puede asegurar hasta que se realicen los experimentos correspondientes.

Literatura citada

Aires, K. R., Santana, A. M., & Medeiros, A. A. (2008). Optical flow using color information: preliminary results. In Proceedings of the 2008 ACM symposium on Applied computing (pp. 1607-1611). ACM.

Adelson, E. H., & Bergen, J. R. (1985). Spatiotemporal energy models for the perception of motion. JOSA A, 2(2), 284-299.

Barron, J. L., Fleet, D. J., & Beauchemin, S. S. (1994). Performance of optical flow techniques. International journal of computer vision, 12(1), 43-77.

Black, M. J. (1992). Robust incremental optical flow (Doctoral dissertation, PhD thesis, Yale university).

Black, M. J., & Anandan, P. (1996). The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. Computer vision and image understanding, 63(1), 75-104.

Darwin, C. (1968). On the origin of species by means of natural selection. 1859. London: Murray.

Dozal, L., Olague, G., Clemente, E., y Hernandez,D. E. (2014). Brain programming for the evolution of an artificial dorsal stream. Cognitive Computation, 6(3):528–557.

Fleet, D. J., & Jepson, A. D. (1990). Computation of component image velocity from local phase information. International journal of computer vision, 5(1), 77-104.

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). Artificial intelligence through simulated evolution.

Gibson, J. J. (1950). The perception of the visual world. Houghton Mifflin.

Gong, M., & Yang, Y. H. (2002). Multi-resolution genetic algorithm and its application in motion estimation. In Pattern Recognition, 2002. Proceedings. 16th International Conference on (Vol. 1, pp. 644-647). IEEE.

Guerra García, Luis Arturo (2016). Atención visual integrando el paradigma de potencialidades en función de la distancia. Tesis de maestría,Centro de Investigación Científica y de Educación Superior de Ensenada, México

Heeger, D. J. (1988). Optical flow using spatiotemporal filters. International journal of computer vision, 1(4), 279-302.

Holland, J. H. (1975). Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence. Ann Arbor, MI: University of Michigan Press.

Horn, B. K., & Schunck, B. G. (1981). Determining optical flow. Artificial intelligence, 17(1-3), 185-203.

Huang, Y., Palaniappan, K., Zhuang, X., & Cavanaugh, J. E. (1995). Optic flow field segmentation and motion estimation using a robust genetic partitioning algorithm. IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(12), 1177-1190.

Kalivas, D. S., & Sawchuk, A. A. (1991). A region matching motion estimation algorithm. CVGIP: Image Understanding, 54(2), 275-288.

Kories, R., & Zimmermann, G. (1986). A versatile method for the estimation of displacement vector fields from image sequences. In IEEE Workshop on Motion: Representation and Analysis (Vol. 101, p. 106).

Koza, J. R. (1992). Genetic programming: on the programming of computers by means of natural selection (Vol. 1). MIT press.

Li, S., Xu, W. P., Wang, H., & Zheng, N. N. (1999). A novel fast motion estimation method based on genetic algorithm. In Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on (Vol. 1, pp. 66-69). IEEE.

Little, J. J., Bulthoff, H. H., & Poggio, T. (1988). Parallel optical flow using local voting. In Unknown Host Publication Title. Publ by IEEE.

Llamas Virgen, Paul (2014). Programación cerebral : un nuevo enfoque para la resolución del problema de seguimiento de objetos en secuencias de imágenes basado en atención visual. Tesis de maestría,Centro de Investigación Científica y de Educación Superior de Ensenada, México

Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision.

Mallot, H. A. (2000). Computational vision: information processing in perception and visual behaviour. MIT Press.

Rechenberg, E. Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. 1973. Frommann-Holzboog Verlag, Stuttgart.

Schwefel, H. P. P. (1993). Evolution and optimum seeking: the sixth generation. John Wiley & Sons, Inc..

Sun, D., Roth, S., & Black, M. J. (2010, June). Secrets of optical flow estimation and their principles. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on (pp. 2432-2439). IEEE.

Sutton, M. A., Wolters, W. J., Peters, W. H., Ranson, W. F., & McNeill, S. R. (1983). Determination of displacements using an improved digital correlation method. Image and vision computing, 1(3), 133-139.

Tagliasacchi, M. (2003, October). Optical flow estimation using genetic algorithms. In International Workshop on Fuzzy Logic and Applications (pp. 309-316). Springer, Berlin, Heidelberg.

Watson, A. B., & Ahumada, A. J. (1985). Model of human visual-motion sensing. JOSA A, 2(2), 322-342.

Zavala-Romero, O., Botella, G., Meyer-Bäse, A., & Base, U. M. (2011, May). Optical flow optimization using parallel Genetic Algorithm. In SPIE Defense, Security, and Sensing (pp. 80581E-80581E). International Society for Optics and Photonics.