Robot autónomo basado en visión computacional, algoritmo genético y computación gráfica para recorrer caminos sin colisión

Autonomous robot based on computer vision, genetic algorithm and computer graphics for collision-free path traversal.

Raul Eduardo Huarote Zegarra1[1], Angel Fernando. Navarro Raymundo[1], y Janett Deisy Julca Flores[1]

[1]Universidad Nacional Tecnológica de Lima Sur, Lima – Perú

RESUMEN

La presente investigación lo que pretende es que el Robot realice los movimientos de acuerdo a una determinada ruta de manera autónoma (con inicio y destino el mismo punto), para lograr ello se basa en visión computacional para encontrar los puntos específicos donde debe de considerar su camino y a la vez optimizar la ruta de camino basándose en algoritmo genético, y para representarlo de manera visual se debe considerar de manera grafica la curva de bezier. La combinación de estos modelos para lograr el objetivo de la presente investigación hace que el robot cumpla con el criterio de un recorrido autónomo, adicionalmente se considera sin colisión, lo cual implica que no se va a retomar un punto ya visitado y/o intersección ya recorrida. Esto se plasma en los siguientes capítulos para poder disgregar este resultado.

Palabras claves: robot, visión computacional, algoritmo genético, computación gráfica, colisión.

ABSTRACT

The present research aims for the Robot to perform movements according to a certain route autonomously (with start and destination at the same point), to achieve this it is based on computer vision to find the specific points where it must consider its path and at the same time optimize the route based on genetic algorithm, and to represent it visually, the Bezier curve must be considered graphically. The combination of these models to achieve the objective of this research makes the robot meet the criterion of an autonomous route, additionally it is considered without collision, which implies that it will not take up a point already visited and/or intersection already traveled. This is reflected in the following chapters to break down this result.

Keywords: robot, computer vision, genetic algorithm, graphics computer, collisions.

INTRODUCCION

El alcance de esta investigación se percibe como beneficioso el logro de la interrelación de estos modelos basados en inteligencia artificial para el campo académico, y se pueda tomar como referencia para futuras investigaciones para aplicarlas en entornos similares. Por tanto, lo que se pretende alcanzar es implementar un robot autónomo basado en visión computacional, algoritmo genético y computación gráfica para recorrer caminos sin colisión.

MATERIALES Y MÉTODOS.

La presente investigación se ha considerado como parte de materiales para la información de la posición del objeto, necesariamente se debe considerar las características de los instrumentos y como se va a adquirir los resultados y los datos que se pueden reportar que representan una distancia.

Para lograr ello es necesario considerar los software y equipos de la Tabla I para poder realizar el funcionamiento de la presente investigación.

Tabla I: Lista de equipos y software necesarios para la realización del proyecto de investigación

DESCRIPCIÓN	CANTIDAD	Detalle de Utilidad
Proteus	1	Necesario para la simulación del prototipo funcional, tanto en la parte del código Arduino, como en la parte de visión artificial con computación gráfica.
Python	1	Lenguaje de programación donde va a realizar el análisis de la optimización, visión computacional y el Arduino.
Kit de carro robot	1	Dispositivo para que evidencie los movimientos y camino generado por los algoritmos.
Arduino Uno	1	Dispositivo que va a realizar el proceso de movimiento del móvil.
Cámara 24Mp o superior	1	Permite encontrar el objeto (de un determinado color) con el fin de genera los puntos de control.

2.1 Objetivos

Para la presente investigación se considera los presentes objetivos:

Objetivo General:

- Recorrer caminos sin colisión mediante un robot autónomo basado en visión computacional, algoritmo genético y computación gráfica.

Objetivos específicos:

- Identificar los puntos característicos o puntos de control mediante visión computacional para la implementación de un robot autónomo basado en visión

computacional, algoritmo genético y computación gráfica para recorrer caminos sin colisión.

- Optimizar la ruta con los puntos de control mediante algoritmo genético para la implementación de un robot autónomo basado en visión computacional, algoritmo genético y computación gráfica para recorrer caminos sin colisión.
- Generar la gráfica de recorrido optimo mediante computación gráfica y curva de bezier para la implementación de un robot autónomo basado en visión computacional, algoritmo genético y computación gráfica para recorrer caminos sin colisión.

2.2 Metodología

La secuencia establecida, para lograr el resultado esperado se grafica en la Figura 1, donde se detalla el proceso de funcionamiento de la presente investigación.

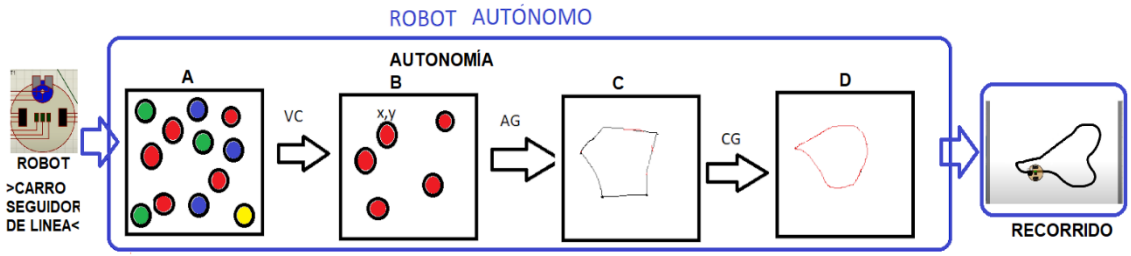

Figura 1. Recorrido del robot en función de la ruta generada por la visión artificial y computación grafica.

2.3 Aplicación del modelo

Para resolver el presente proyecto se ha disgregado en diferentes campos de la inteligencia artificial, el cual cada uno cumple un rol especifico hasta poder lograr el objetivo general que es la autonomía del robot, por tanto se hará la explicación de cada uno de ellos en los siguientes puntos:

A. Visión computacional: Se aplica la visión computacional para poder identificar los diferentes obstáculos (llamados puntos de control) encontrados en diferentes posiciones (aleatorias), el cual es percibido (captado) mediante cámara de regulares características, los obstáculos son encontrados de acuerdo a un determinado color (lo escoge el usuario) por tanto este resultado de haber encontrado los diferentes obstáculos representados en posiciones en un plano (ver Figura 2), implica que estas posiciones van a ser tomadas como fuente para poder generar posteriormente un determinado recorrido.

Para lograr encontrar un color especifico de acuerdo a la escena, indistintamente de la forma del objeto, debemos tener en cuenta el rango de colores (ejemplo rojo, rojo alto u oscuro a rojo claro), del cual es necesario obtener en base a la posición central del pixel, esto equivaldría a la coordenada o posición en el plano, cambe resaltar que esta posición encontrada, se realizara de manera análoga con la cámara para poder encontrar objetos de diferentes colores, el cual será tomado como los puntos de control. Para lograr esto se ha utilizado una librería de libre uso y disponible en la web que es el OpenCV, que va a ser invocado por otro software de características libre que es el Python. Así obtener el

objetivo planteado. EL código fuente se presenta en la Figura 3. Donde se encuentra a partir de los datos originales de la Figura 2.
Para lograr esto se basó en la fuente de (Enrique, 2007) el cual menciona que visión computacional “es el estudio de estos procesos, para entenderlos y construir máquinas con capacidades similares”.

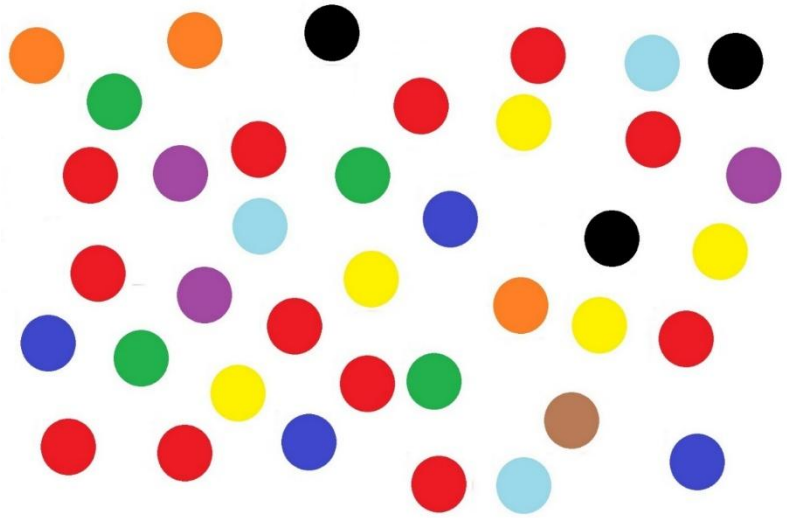

Figura 2. Imagen original en el cual se desea encontrar las coordenadas de acuerdo con las posiciones de los colores.

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

from Punto import Punto
from Curva import Curva
from AlgoritmoGenetico import Ciudad, AlgoritmoGenetico

# Obtener puntos de control de una imagen
imagen = cv2.imread("puntos.jpg")
hsv = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)

rojo_bajo = np.array([175, 100, 20])
rojo_alto = np.array([179, 255, 255])

mascara_rojo = cv2.inRange(hsv, rojo_bajo, rojo_alto)
cv2.imshow("mascara rojo", mascara_rojo)

cnts, _ = cv2.findContours(
    mascara_rojo, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Calculado coordenas de los puntos
ciudades = []
ci = 0
for c in cnts:
    epsilon = 0.01*cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c, epsilon, True)
    if len(approx) > 5:
        x, y, w, h = cv2.boundingRect(approx)
        # print(x, y, w, h)
        ciudades.append(Ciudad(Punto(x, y), "P"+str(ci)))
        ci += 1
        cv2.putText(imagen, "("+str(x)+","+str(y)+")",
                    (x, y-5), 1, 1, (0, 0, 0), 1)
cv2.imshow('coordenadas de imagen', imagen)
```


DOI: https://doi.org/10.52248/eb.Vol4Iss3.179

```python
import numpy as np

class Punto:
    def __init__(self, x, y):
        self.p = np.array([x, y, 1])

    def P(self):
        return self.P

    def value(self):
        return np.array([self.p[0], self.p[1]])

    def T(self, Tx, Ty):
        T = np.array([[1, 0, 0],
                      [0, 1, 0],
                      [Tx, Ty, 1]])
        self.p = np.dot(self.p, T)

    def R(self, angulo):
        angulo = angulo * np.pi / 180
        R = np.array([[np.cos(angulo), np.sin(angulo), 0],
                      [-np.sin(angulo), np.cos(angulo), 0],
                      [0, 0, 1]])
        self.p = np.dot(self.p, R)

    def ejeX(self):
        Rx = np.array([[1, 0, 0],
                       [0, -1, 0],
                       [0, 0, 1]])
        self.p = np.dot(self.p, Rx)

    def ejeY(self):
        Ry = np.array([[-1, 0, 0],
                       [0, 1, 0],
                       [0, 0, 1]])
        self.p = np.dot(self.p, Ry)

    def ejeD(self):
        R = np.array([[-1, 0, 0],
                      [0, -1, 0],
                      [0, 0, 1]])
        self.p = np.dot(self.p, R)

```

Figura 3. Código fuente en Python y librería de OpenCv, para obtener las coordenadas de acuerdo con el color de un objeto.

Posiciones especificas: Como resultado de aplicar el código fuente, se ha obtenido como parte del resultado la máscara (Figura 4), necesario para poder identificar el obstáculo de un determinado color en particular. Para lograr ello se basó en la fuente de (Morales, 2011) donde considera al procesamiento digital de imágenes como “los sistemas de visión artificial o de visión por computadora, terminología en la actualidad de uso muy habitual, tratan de englobar un conjunto de procedimientos relacionados con el procesamiento y análisis digital de imágenes, los cuales abarcan un sinfín de técnicas y herramientas matemáticas, físicas, computacionales y de ingeniería con aplicaciones en numerosos campos de la vida moderna”. También para (Gutiérrez, 2003) procesamiento digital de imágenes es la percepción humana “el ser humano utiliza imágenes digitales, bien para guardarlas o para modificarlas, como teledetección, imágenes médicas o fotográficas, etc.”. Por tanto, considera (Gutiérrez, 2003) para las imágenes digitales que “Proceden del muestreo espacial y en intensidad de la imagen óptica. Están formadas por una matriz de elementos (píxeles)”.

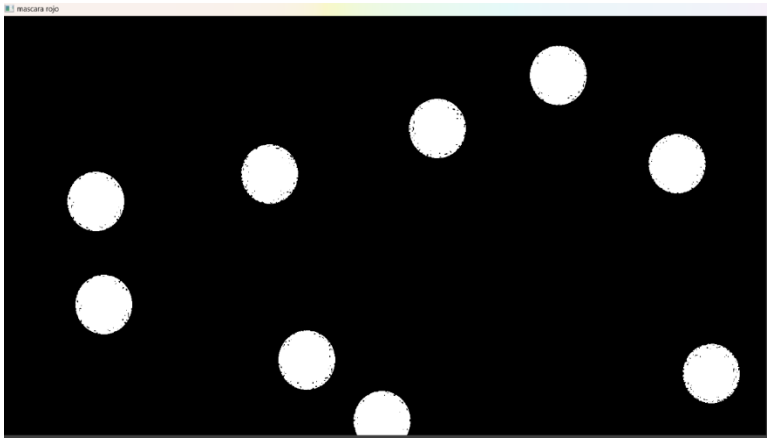

Figura 4. Imagen de la máscara de color encontrado en función de un color determinado.

Aplicando este modelo se logra ubicar los puntos en el plano, en referencia a la posición inferior derecha como posición inicial (0,0), para así encontrar de acuerdo con el color (este caso el de color rojo o cercano a ello según tonalidad), y estos ser tomados como puntos de control, tal como se visualiza en la Figura 5.

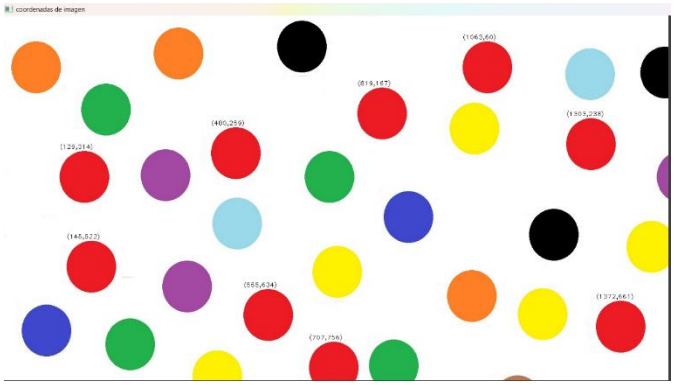

Figura 5. Imagen en el cual se ha aplicado el código fuente en Python y OpenCv, para lograr encontrar las diferentes posiciones en el plano

B. Algoritmo genético: Aplicando este modelo basado en inteligencia artificial, es identificar la ruta más corta a partir de los puntos de control, basándose en algoritmos genéticos. Se ha considerado los puntos de control, de acuerdo con los valores de las posiciones encontradas en el párrafo anterior. Estas posiciones están en el plano y que la distancia existente entre ellas hace que se obtenga una matriz de distancias, y que a partir de esos datos va a ser insumo para poder encontrar la ruta corta basado en algoritmo genético.

Considerando para ello la referencia de su publicación la representación de Charles Darwin (Darwin, 1859) donde “sobrevive el más apto”. Así después la representación en software fue dada por Bremermann (Bremermann, 1962) y que posteriormente fue utilizada por Holland (Holland, 1992), este autor acuñó el nombre de “Algoritmos Genéticos”, posteriormente Golberg (olberg, 1989) toma esta idea como “métodos adaptativos", generalmente usados en problemas de búsqueda y optimización de parámetro. Hoy es una técnica basada en la teoría de adaptación de Darwin que no es más que una adaptación del biólogo (Lamarck, 1801), basados en la reproducción sexual y en el principio de supervivencia del más apto.

Por tanto, algoritmo genético es un modelo computacional de búsqueda de la posible mejor solución, basado en la adaptación del modelo evolutivo.
Datos de configuración del algoritmo genético:
Los datos son el tamaño de la población, en el cual se ha considerado con 50, ya que de acuerdo con la cantidad de puntos y las pruebas realizadas se obtiene resultados con este valor. También se ha considerado la probabilidad de cruce (Pc) el cual permite ir mejorando de acuerdo con los mejores individuos de cada generación, propio del algoritmo genético, por tanto, se ha colocado un valor de 0.95 ya que el valor alto implica una buena posibilidad de cruce de acuerdo con los individuos seleccionado. Con respecto a la probabilidad de mutación (Pm) se ha considerado un valor de 0.2 el cual es un número bajo, considerando que si bien se requiere un cambio en proceso (para aplicar la variabilidad) esto no necesariamente tiene que ser un valor alto, ya que perdería cierta uniformidad en el proceso. Respecto al número de generaciones que se ha aplicado, se considera un valor de 100 ya que esto permite encontrar una posible solución, con las restricciones puestas.
Esto se ha plasmado en el código fuente presentado en la Figura 6. Donde para mostrar los resultados de los puntos y la ruta corta se requiere el código presentado en la Figura 7. Mostrando los resultados ya listos para que el dispositivo robot pueda moverse. Por tanto, se ha logrado realizar el objetivo de generar la ruta corta a partir de puntos de control de una determinada escena.

```
# Aplicando algoritmo genetico
tampoblacion = 50
Pc = 0.95
Pm = 0.2
numgeneraciones = 100

AG = AlgoritmoGenetico(tampoblacion, numgeneraciones, Pc, Pm, ciudades)
sol = AG.run()  # Obtenemos mejor solucion

ruta = []
for p in sol.ruta:
    ruta.append(p.pto)
# Camino cerrado
ruta.append(sol.ruta[0].pto)  # Añadimos el primer punto

# Calculamos los Ais
A = []
for i in range(len(ruta)-1):
    [x1, y1] = ruta[i].value()
    [x2, y2] = ruta[i+1].value()
    A.append(Punto((x1+x2)/2, (y1+y2)/2))

# Calculamos los Bis
B = []
for i in range(len(A)-1):
    [x1, y1] = ruta[i].value()
    [x2, y2] = ruta[i+1].value()
    [x3, y3] = ruta[i+2].value()
    d12 = ((x2-x1)**2 + (y2-y1)**2)**0.5
    d23 = ((x3-x2)**2 + (y3-y2)**2)**0.5
    r = d12/d23
    x = (A[i].value()[0] + (r*A[i+1].value()[0]))/(1+r)
    y = (A[i].value()[1] + (r*A[i+1].value()[1]))/(1+r)
    B.append(Punto(x, y))
[x1, y1] = ruta[-2].value()
[x2, y2] = ruta[0].value()
[x3, y3] = ruta[1].value()
d12 = ((x2-x1)**2 + (y2-y1)**2)**0.5
d23 = ((x3-x2)**2 + (y3-y2)**2)**0.5
r = d12/d23
x = (A[-1].value()[0] + (r*A[0].value()[0]))/(1+r)
y = (A[-1].value()[1] + (r*A[0].value()[1]))/(1+r)
B.append(Punto(x, y))
```

Figura 6. Obtener los puntos de control que son las posiciones de las imágenes, representada en función de la distancia euclidiana.

Los códigos siguientes han sido necesario para crear las diferentes clases y poder encontrar la ruta corta: Encontrando la mejor solución en el siguiente conjunto de puntos tal como se muestra en la Figura 7, generando para este caso los puntos P0 – P4 – P9 – P11 – P10 – P1 – P2 – P6 – P7 – P8 – P5 – P3. Indicando que inicia en el punto 0, de ahí al punto 4, si sucesivamente hasta el punto 3.

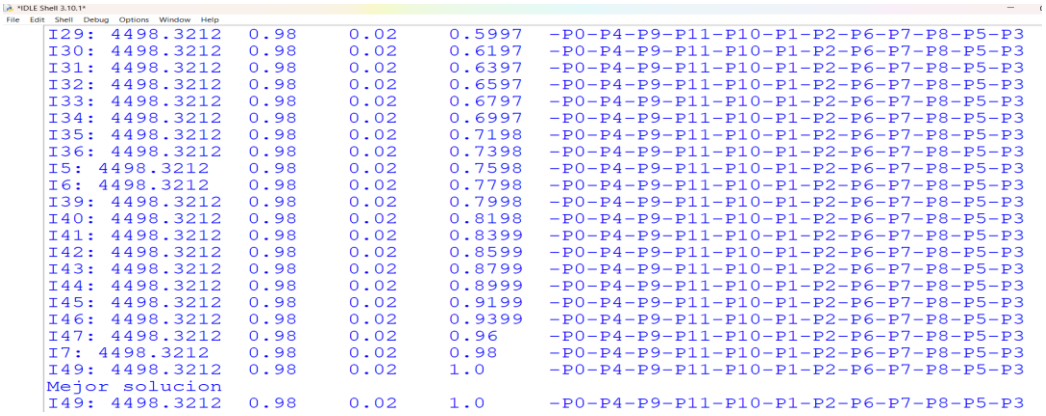

Figura 7. Ejecución del algoritmo genético para encontrar la ruta corta de acuerdo a los puntos de control.

C. Generar curva: Para la generación de la curva, considerando que sea lo más suave posible en el recorrido del robot, por eso es necesario considerar una de los diferentes modelos, a pesar de que existen diferentes modelos para generar curvas, como b-spline, hermite, cubica, etc, se ha considerado el modelo de Bezier, ya que estos modelos no permiten pasar por los puntos de control a comparación de Bezier, por tal motivo ha sido necesario aplicar este modelo, tal como se identifica en la Figura 8, y se refleja el resultado en la Figura 9.

Se tomo como fuente bibliográfica para resolver este punto a (Donald P. Hearn, 2006), así también a (Escribano, 1995) y (Olivares, 2002), permitiendo cumplir el objetivo específico planteado en este punto.

```
80
81  # Puntos de control para cada curva
82  curvas = []
83  for i in range(len(ruta)-1):
84      ptsct = []
85
86      ptsct.append(ruta[i])
87
88      [xB, yB] = B[i-1].value()
89      Tx = ruta[i].value()[0] - xB
90      Ty = ruta[i].value()[1] - yB
91      Ai = Punto(A[i].value()[0], A[i].value()[1])
92      Ai.T(Tx, Ty)
93      ptsct.append(Ai)
94
95      [xB, yB] = B[i].value()
96      Tx = ruta[i+1].value()[0] - xB
97      Ty = ruta[i+1].value()[1] - yB
98      Ai = Punto(A[i].value()[0], A[i].value()[1])
99      Ai.T(Tx, Ty)
100     ptsct.append(Ai)
101
102     ptsct.append(ruta[i+1])
103
104     curvas.append(Curva(ptsct))
105
106 fig, ax = plt.subplots()
107 fig.set_size_inches(9, 7.5)
108 fig.set_dpi(100)
```

Figura 8. Código fuente para mostrar los puntos y la ruta usando la curva de bezier

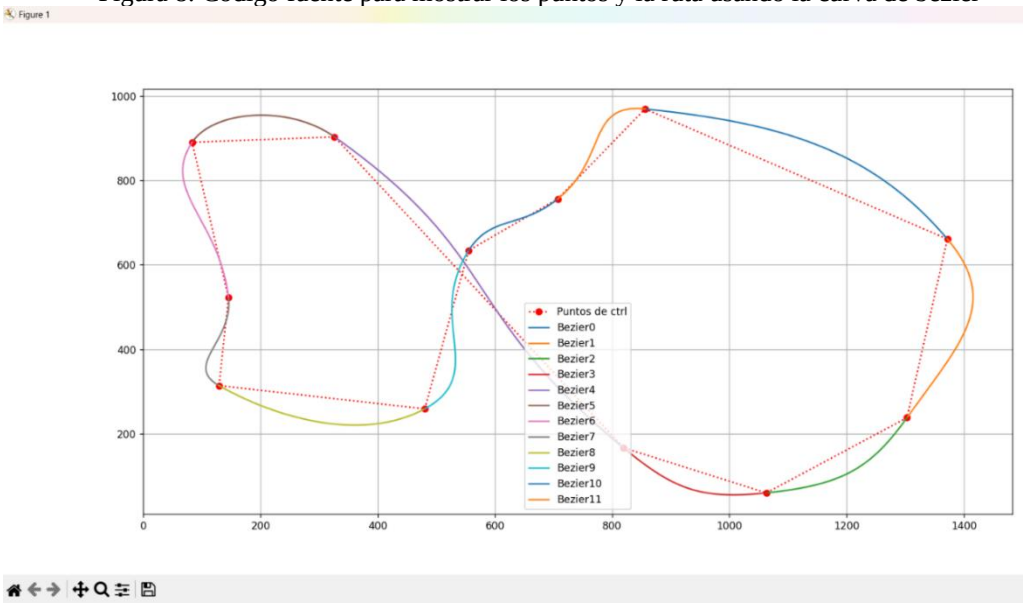

Figura 9. Cuál sería el recorrido del móvil a partir de los puntos de control y la ruta obtenida aplicando el algoritmo genético.

D. Recorrido: Comprobación del sistema que llega a recorrer de manera autónoma los caminos datos evadiendo los obstáculos. Para lograr recorrer de manera autónoma, cabe resaltar que la autonomía está contemplada en los párrafos anteriores (basado en visión artificial, algoritmo genético y computación grafica), por tanto, para el presente informe se centra específicamente en la simulación de la parte hardware donde se va a considera un robot (carro) en el cual tendrá que seguir una línea generada, para ello en el robot se tiene que considerar los movimientos de las llantas para poder recorrer una determinada ruta. El criterio de evadir los obstáculos se centra en el modelo utilizado para poder

recorrer los puntos, específicamente se ha usado la generación de curva de Bézier, ya que este tipo de curva genera una ruta basada en los puntos de control, pero no pasa por los puntos de control, indicando que evade los obstáculos, pero a la vez da un camino a recorrer. Por tanto, como se tiene una ruta ya generada, tal como se muestra en la Figura 10. Así a su vez para poder ser usado desde el entorno Arduino, se muestra el código fuente en Arduino en la Figura 11 para lograr el resultado. Por tanto, en la Figura 12, se encuentra el funcionamiento del recorrido del robot para poder realizar el respectivo seguimiento de la línea generada.

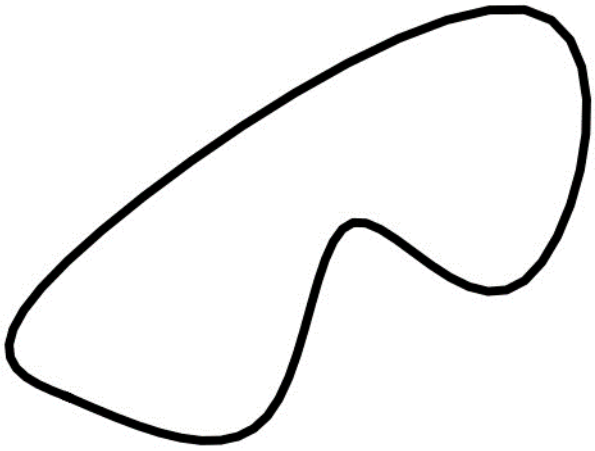

Figura 10. Ruta generada por algoritmos basado en visión artificial y computación gráfica, para poder ser recorrida por el robot.

```
#include <Servo.h>
#define Trigger 3
#define Echo  4
// Define las llantas
#define LlantaIA  7
#define LlantaIR  6
#define LlantaDA  5
#define LlantaDR  4
//Define los sensores
#define Si 13
#define Sc 12
#define Sd 11
//variables
int sA,sB,sC;
int velocidad = 0;
int Estabilizador = 0;
int Graduador,Referencia;
unsigned long Tiempo;
int Centraliza = 0;
int Direccion = 0;
//velocidad
#define PwmI 10
#define PwmD 9
Servo jkr;
int Distancia;
void setup(){
  //para el sensor ultrasonido
  pinMode(Trigger,OUTPUT);
  pinMode(Trigger,OUTPUT);
  pinMode(Echo,INPUT);
  pinMode(A0,INPUT);
  //para las llantas
  pinMode(LlantaIA,OUTPUT);
  pinMode(LlantaIR,OUTPUT);
  pinMode(LlantaDA,OUTPUT);
  pinMode(LlantaDR,OUTPUT);
  //para los sensores
  pinMode(Si,INPUT);
  pinMode(Sc,INPUT);
  pinMode(Sd,INPUT);
  //configuracion pwm
  pinMode(PwmI,OUTPUT);
  pinMode(PwmD,OUTPUT);
  //Velocidad de comunicacion
  Serial.begin(9600);
  //jkr.attach(3);
  //jkr.write(90);
  }
  void loop(){
    // Leer los sensores
    sA = digitalRead(Si);
    sB = digitalRead(Sc);
    sC = digitalRead(Sd);
    //Sensor Central
    if (sA==LOW && sB==HIGH && sC==LOW){
      analogWrite(PwmI,100);
```

```
      analogWrite(PwmI,100);
      analogWrite(PwmD,100);
      digitalWrite(LlantaIA,HIGH);
      digitalWrite(LlantaIR,LOW);

      digitalWrite(LlantaDA,HIGH);
      digitalWrite(LlantaDR,LOW);

      Referencia = 0;
      Graduador = 0;
      Estabilizador = 0;
      delay(300);
      Direccion = 0;
    }
    //Sensor central izquierda
    if (sA==HIGH && sB==HIGH && sC==LOW){
      analogWrite(PwmI,50);
      analogWrite(PwmD,150+Graduador);
      digitalWrite(LlantaIA,HIGH);
      digitalWrite(LlantaIR,LOW);

      digitalWrite(LlantaDA,HIGH);
      digitalWrite(LlantaDR,LOW);

      delay(100);
      analogWrite(PwmI,30);
      analogWrite(PwmD,100+Graduador);
    }
    //Sensor izquierda
    if (sA==HIGH && sB==LOW && sC==LOW){
      analogWrite(PwmI,15);
      analogWrite(PwmD,140 + Graduador);
      digitalWrite(LlantaIA,HIGH);
      digitalWrite(LlantaIR,LOW);

      digitalWrite(LlantaDA,HIGH);
      digitalWrite(LlantaDR,LOW);
      Estabilizador = 1;
    }
    // SensorDerecha
    if (sA==LOW && sB==LOW && sC==HIGH){
      analogWrite(PwmI,140 + Graduador);
      analogWrite(PwmD,15);
      digitalWrite(LlantaIA,HIGH);
      digitalWrite(LlantaIR,LOW);

      digitalWrite(LlantaDA,HIGH);
      digitalWrite(LlantaDR,LOW);
      Estabilizador = 2;
    }
    //Estabilizada
    if (sA==HIGH && sB==HIGH && sC==HIGH){
      if (Estabilizador == 1){
        analogWrite(PwmI,80);
        analogWrite(PwmD,100);
        digitalWrite(LlantaDA,HIGH);
        digitalWrite(LlantaDR,LOW);
        Referencia = 0;
        Graduador = 0;
      }
      if (Estabilizador == 2){
        analogWrite(PwmI,200);
        analogWrite(PwmD,20);
        digitalWrite(LlantaIA,HIGH);
        digitalWrite(LlantaIR,LOW);
        digitalWrite(LlantaDA,HIGH);
        digitalWrite(LlantaDR,LOW);
        Referencia = 0;
        Graduador = 0;
      }
    }
  }

      analogWrite(PwmD,100+Graduador);

      digitalWrite(LlantaIA,HIGH);
      digitalWrite(LlantaIR,LOW);
      digitalWrite(LlantaDA,HIGH);
      digitalWrite(LlantaDR,LOW);
    }
    //Sensor central derecha
    if (sA==LOW && sB==HIGH && sC==HIGH){
      analogWrite(PwmI,150+Graduador);
      analogWrite(PwmD,50);
      digitalWrite(LlantaIA,HIGH);
      digitalWrite(LlantaIR,LOW);

      digitalWrite(LlantaDA,HIGH);
      digitalWrite(LlantaDR,LOW);

      Referencia = 0;
      delay(100);
      analogWrite(PwmI,100 + Graduador);
      analogWrite(PwmD,30);

      digitalWrite(LlantaIA,HIGH);
      analogWrite(LlantaIR,LOW);
      digitalWrite(LlantaDA,HIGH);
      digitalWrite(LlantaDR,LOW);
      Referencia = 0;
    }
      analogWrite(PwmD,100);
      digitalWrite(LlantaIA,HIGH);
      digitalWrite(LlantaIR,LOW);
      digitalWrite(LlantaDA,LOW);
      digitalWrite(LlantaDR,HIGH);
      Referencia = 0;
      Graduador = 0;
      delay(700);
    }
    if (Estabilizador == 2){
      analogWrite(PwmI,100);
      analogWrite(PwmD,80);
      digitalWrite(LlantaIA,LOW);
      digitalWrite(LlantaIR,HIGH);
      digitalWrite(LlantaDA,HIGH);
      digitalWrite(LlantaDR,LOW);
      Referencia = 0;
      Graduador = 0;
      delay(700);
    }
  }
  if (sA==LOW && sB==LOW && sC==LOW){
    if (Estabilizador == 1){
      analogWrite(PwmI,20);
      analogWrite(PwmD,200);
      digitalWrite(LlantaIA,HIGH);
      digitalWrite(LlantaIR,LOW);
      digitalWrite(LlantaDA,HIGH);
```

Figura 11. Código fuente en el lenguaje Arduino para poder realizar el movimiento del robot.

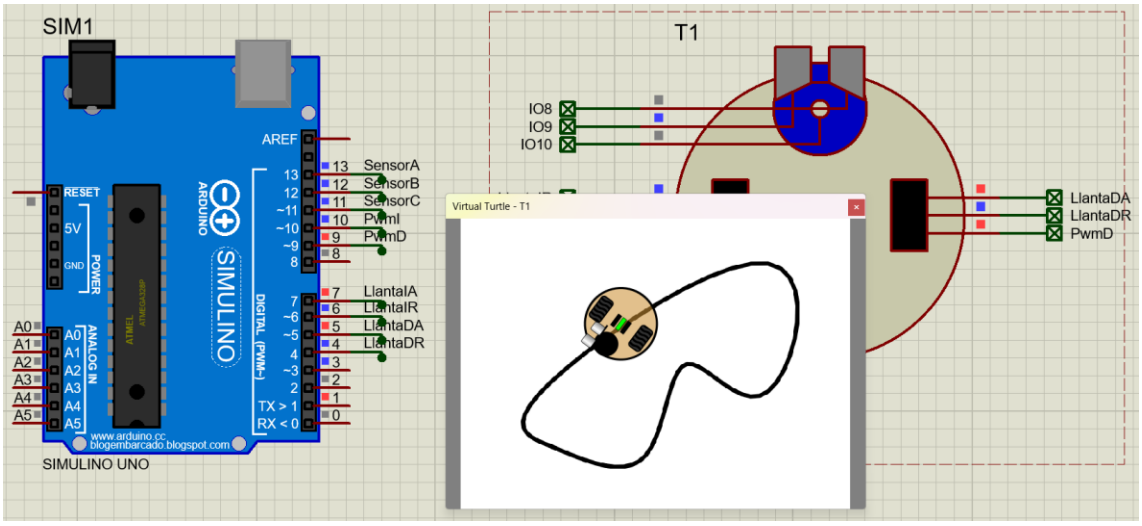

Figura 12. Recorrido del robot en función de la ruta generada por la visión artificial y computación grafica en un entorno simulado Proteus.

RESULTADOS

Si tomamos como parte inicial cada una de las acciones se detalla en la Figura 1, como parte del recorrido ***autónomo*** del ROBOT, tiene que realizar el seguimiento de línea representado en entorno virtual (Proteus) cumplir cada uno de los criterios:

Visión computacional (A): Imagen original en el cual representa una escena real, donde encuentra diferentes objetos en el cual está presto a identificar a varios de un solo tipo (un solo color).

Posiciones especificas (B): Identifica a los objetos de un solo color (en el ejemplo del gráfico está de color rojo) y encuentra las posiciones de cada uno de ellos, el cual se ubica dichas posiciones usando VC (Visión computacional).

Por tanto, se ha cumplido el primer objetivo específico que es "Identificar los puntos característicos o puntos de control mediante visión computacional para la implementación de un robot autónomo basado en visión computacional, algoritmo genético y computación gráfica para recorrer caminos sin colisión".

Algoritmo genético (C): Necesariamente para saber cómo recorrer tiene que encontrar la ruta más corta, para ello necesariamente tiene que usar un método de optimización, como es el AG (algoritmo genético) y generarle los puntos a recorrer.

Por tanto, se ha cumplido el segundo objetivo específico que es "Optimizar la ruta con los puntos de control mediante algoritmo genético para la implementación de un robot autónomo basado en visión computacional, algoritmo genético y computación gráfica para recorrer caminos sin colisión".

Generar curva (D): Una vez encontrada la ruta más corta por medio de los puntos a recorrer, tiene que generar el camino, para ello se tiene que agenciar de un algoritmo de CG (Computación Grafica) como es Bezier para genera la línea en un archivo .png.

Para el recorrido autónomo del robot ya puede realizar el recorrido idóneo, puesto que ya tiene el camino definido (en un .png) y esto se ve evidenciado en el entorno Proteus.

Por tanto, se ha cumplido el tercer objetivo específico que es "Generar la gráfica de recorrido optimo mediante computación gráfica y curva de bezier para la implementación de un robot autónomo basado en visión computacional, algoritmo genético y computación gráfica para recorrer caminos sin colisión".

Recorrido (E): esto implica el resultado de la presente investigación, por tanto, se contempla el cumplimiento de la parte autónoma del robot, ya que cuenta con los criterios necesarios para poder realizar el recorrido, a partir de la imagen o escena. Implicando que se ha cumplido el objetivo general "Recorrer caminos sin colisión mediante un robot autónomo basado en visión computacional, algoritmo genético y computación gráfica".

DISCUSIÓN.

Las investigaciones realizadas por otros autores, donde consideran otros modelos para obtener la ruta mas corta como algoritmo hamiltoniano, algoritmo de las 8 reinas, Dijkstra. Considerando que para la presente investigación se ha tomado un método alternativo para encontrar la ruta mas corta, como es en este caso el algoritmo genético, adicionalmente a ello se ha considerado en proceso de optimización la restricción de que no se crucen en el recorrido.

La implementación del robot en físico se ha realizado posterior a la evaluación previa del uso de simuladores como proteus.

Se puede utilizar el código fuente de la presente investigación para futuras investigaciones, donde implique tener cierta autonomía para el recorrido de un determinado objeto teniendo obstáculos que considerar.

Demora en la obtención de la escena de acuerdo con el tipo de color de objeto, considerando el fondo ya que no es uniforme, por lo tanto, se tomará como referencia el código presentado.

Demora en la obtención del recorrido del robot en la curva generada, ya que se tenía que realizar diferentes pruebas por el grosor de la línea y las intersecciones que se puedan encontrar, por lo tanto, se tomará como referencia el código presentado.

CONCLUSIÓN.

Se puede comprobar que se puede aplicar el modelo basado en cada una de las áreas de la inteligencia artificial, para poder resolver y aplicar en entornos donde se necesita encontrar la ruta más corta, considerando que no debe haber colisión en su recorrido.

Los resultados obtenidos correspondientes a los datos de la realización del prototipo funcional, por tanto, se pueden representar un recorrido, sin colisiones existentes.

La presente investigación se puede tomar como base para futuras investigaciones, ya que se puede tomar estos insumos la aplicación de casos en donde se requiera optimizar ruta, así como para considerar las restricciones.

REFERENCIAS BIBLIOGRÁFICAS

Bremermann, H. J. (1962). Optimization through evolution and recombination. Washinton, D. C: In Self-organization system, M. C. Yovitts et al. Spartan Books.

Darwin, C. (1859). El origen de las especies. London College of Cambridge: London College of Cambridge.

Donald P. Hearn, P. B. (2006). Graficas por Computadora con OpenGL. Estados Unidos: Ed.Prentice-Hall Hispanoamericana.

Escribano, M. (1995). Programación de Gráficos en 3D. Estados Unidos: Iberoamericanal.

Gutiérrez, E. A. (2003). PROCESAMIENTO DIGITAL DE IMAGEN. España: Universidad de León.

Holland, J. (1992). Adaptation in Natural and Artificial Systems. Cambridge: MIT Press.

Lamarck. (1801). Système des animaux sans vertèbres. Francia: Francia.

Morales, R. R. (2011). Procesamiento y analisis digital de imagenes. Mexico: RA-MA.

olberg, D. (1989). Genetics Algorithms in search, optimization and machine learning. EEUU: Addison-Wesley Professional.

Olivares, M. (2002). Curvas Fractales. Estados Unidos: RJ, Alfaomega.