

Proyecto de Visión para Robots (TC-3050)

**Revisión: Tercera y Etapa Final**

**Técnicas de Visión Computacional y de Planeación de Caminos**

**Profesor: Dr. José Luis Gordillo**

**Equipo 4:**

Melchor Daniel Ramos Ibarra A01282881  
Rafael Alonso Ugarte Rojas A00820677  
Antonio Enrique Franco Gámez A01252180  
Yelanetzi Zahylí Chávez Hernández A00820098

**Resumen:**

La tercera etapa de este proyecto, al ser la etapa final, se trata de desarrollar un sistema de planeación y seguimiento de caminos, incluyendo un módulo para la interpretación de la imagen que se utilizara para crear el espacio de trabajo, obteniendo de esta los obstáculos y el espacio libre. Para este caso se utilizará la imagen de un estacionamiento y se utilizará un objeto punto con radio fijo para representar un automóvil y los obstáculos se ensancharon de acuerdo a este radio. El punto final será definido por el usuario y se utilizará la etapa dos del proyecto para definir el punto y dirección de inicio. Después se generará un camino del punto de inicio al punto final y se representará tanto el camino, cómo una demostración del avance del objeto.

**Introducción:**

La planeación y seguimiento de caminos se refiere a una serie de procesos, mediante los cuales, se analiza una imagen para identificar un espacio de trabajo, dentro del cual existe un espacio libre y obstáculos, esto se obtiene mediante el procesamiento de la imagen de espacio de trabajo, binarización y clasificando el espacio libre cómo una región y los obstáculos cómo otra. También es necesario el encontrar el radio del objeto que realizará el movimiento, para representarlo cómo un punto y ensanchar los obstáculos para que tengan el tamaño adecuado.

Con el espacio de trabajo obtenido, es necesario calcular una función que conecte el punto inicial *pi* y el punto final *pf*. Para esto, se debe realizar una expansión de propiedades desde *pi* hasta *pf*, y calcular esta función sobre el espacio libre. Para esto existen múltiples métodos: cómo los Caminos de Probabilidades, los Campo de Potencial y el Algoritmo NF1. Con la función definida, se puede planear un camino a seguir para el robot objeto.

En esta etapa final del proyecto, se realizará un programa que realice todos estos procesos: procesamiento de trabajo (definición de espacio libre y obstáculos), abstracción y generación del grafo, búsqueda en el grafo, y proyección geométrica de la solución. Para esto se utilizarán las etapas previas del proyecto y los conocimientos adquiridos en la parte final del semestre.

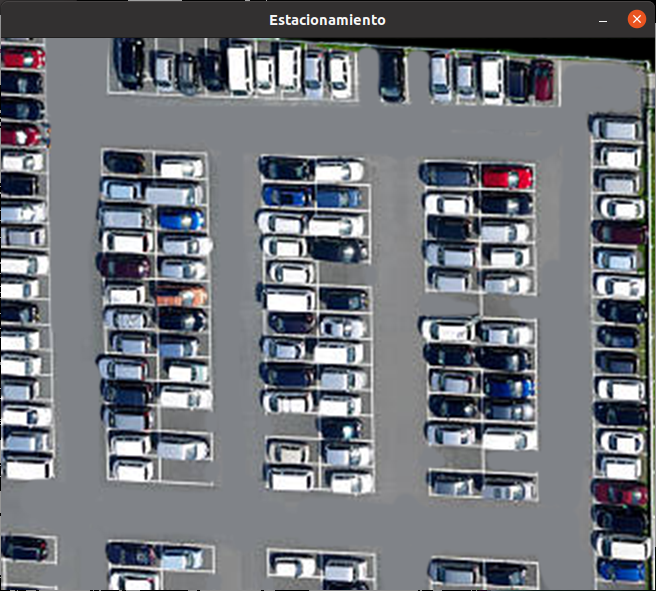
**Desarrollo teórico y experimental:**

Al igual que en la etapa anterior, enfocada en el reconocimiento y clasificación de imágenes, el programa completo de planeación y seguimiento de rutas también está basado en dos procesos clave: la calibración del color, y el entrenamiento de los objetos a reconocer.

Para la planeación de rutas, se utilizó la técnica de campos de potencial, creados con base en el algoritmo de gota de aceite. Sin embargo, para poder aplicar este algoritmo a una imagen fue necesario refinar y utilizar tecnicas de conversión entres espacios de color, uso de filtros paso bajos, binarización de una imagen y funciónes morfológicas.

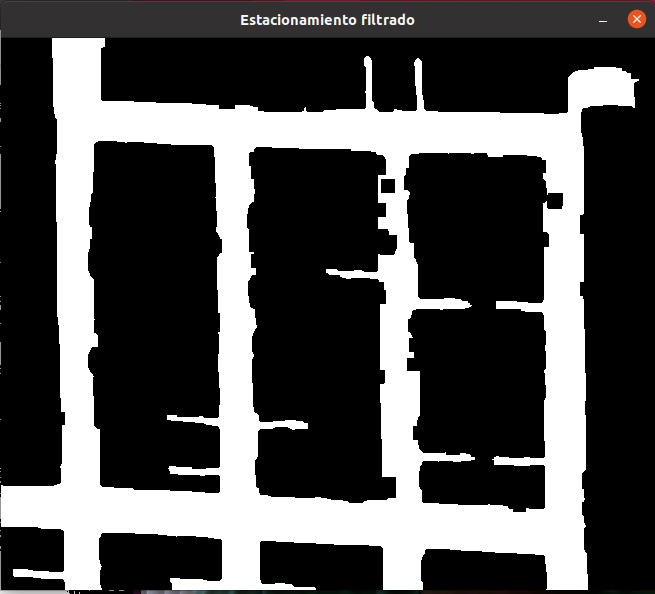
**Capturación de imagen**

Al momento de utilizar la imagen fue necesario hacer unos pequeños ajustes para reducir la imagen del estacionamiento y que no hubiera tanto “ruido” en la imagen, en la figura 1 se muestra la imagen original que se introduce para después hacer la binarización y definir bien los espacios de estacionamientos sobre el camino para llegar a ellos.

  
**Figura 1.** Imagen original

**Definición de obstáculos y radio del carro**

Después de procesar la imagen original se hace la binarización con unos pequeños retoques usando funciones de erosión y dilatación las cuales nos arrojan una imagen en la cual se pueden diferenciar fácilmente las secciones donde hay espacios de estacionamiento libres, todo esto se muestra en la figura 2.

  
**Figura 2.** Imagen binarizada y erosionada.

Para terminar con este segundo punto se seleccionó un radio del carro el cual es de 12 pixeles, seleccionamos esta medida debido al espacio que hay para estacionar un carro, por la calidad de la imagen esos fueron los pixeles necesarios para que la imagen pudiera entrar en los espacios sin problema, en la figura 3 se puede ver un cuadro azul el cual representa el robot que después se movería hasta llegar a un espacio de estacionamiento libre.

  
**Figura 3.** Imagen original con “carro” en color azul

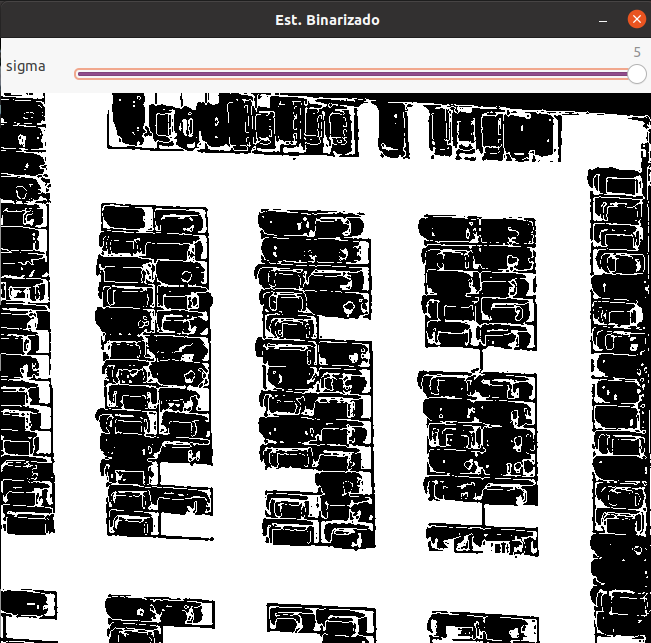
**Binarización y ensanchado de los obstáculos.**

Una vez que capturamos la imagen e hicimos la binarización con las técnicas vistas en parciales anteriores, recurrimos a ensanchar los obstáculos. Nuestro objeto es el área gris así que todo lo demás representa el área a ser erosionada o ensanchada.



**Figura 4.** Selección del objeto a mantener en la imagen.

Una vez obteniendo la binarización escogemos un sigma adecuado para trabajar con los obstáculos, en nuestro caso es un valor de 5. Sin embargo, dadas las condiciones de la edición del camino del estacionamiento y el hecho de que no siempre son los mismos colores y hay interferencias como sombras, relieves, etc., tuvimos que recurrir a un filtro mediado para eliminar la sal y pimienta de la imagen. Después de este proceso la imagen binarizada termina siendo muy clara en cuanto a los caminos.



**Figura 5.** Imagen filtrada y binarizada.

Para ensanchar los obstáculos recurrimos a la técnica de erosión, ya que era la que nos permitía disminuir/erosionar los caminos, que son nuestro objeto, para evitar que el radio del coche colisionara con las paredes del estacionamiento.



**Figura 6.** Estacionamiento después del filtro de erosión con un kernel de 7x7.

**Definición del punto de llegada con un click a un estacionamiento válido.**

Para definir los puntos de llegada fue necesario obtener las coordenadas de los cajones de estacionamiento. Se optó por generar primero los vértices con la función que genera las coordenadas al dar click en la pantalla y eso nos dió los rangos precisos dentro de los cuales el estacionamiento puede ser válido.



**Figura 7.** Espacios de estacionamiento válidos.

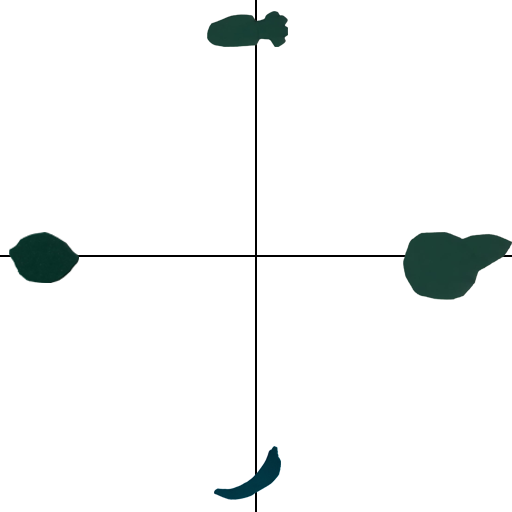
Se creó una función que es llamada cuando se le da click a la imagen erosionada y filtrada la cual nos dice si el espacio seleccionado está dentro de los rangos de coordenadas de los cajones de estacionamiento válidos. Si es válido, dibuja el objetivo dentro del cuadro para estacionar en la coordenada que le corresponde, la más cercana, y lo marca con un punto, se almacena una variable que posteriormente será utilizada como punta de llegada.



**Figura 8.** Puntos finales en los cajones de estacionamiento válidos.

**Definición de punto de entrada**

Para la definición del punto de entrada del camino, se utilizó el programa creado en la etapa dos del proyecto, sin embargo, primero fue necesario realizar cambios en cuanto a las figuras, ya que tuvimos problemas al momento de distinguirlas en la etapa previa. Para esto se sustituyeron 3 de las 4 figuras: La pera pasó a ser una zanahoria, la sandía a ser una banana y la manzana a ser una naranja.



**Figura 9.** Nueva mira de reconocimiento.

Con este cambio a las figuras, se logró identificar correctamente cada cuadrante, los cuales corresponden a su vez con los cuatro puntos de entrada posibles, también se utilizó la orientación de las figuras largas (eje y en la mira) para determinar la dirección inicial del movimiento en el camino, si la figura está vertical, el movimiento es hacia arriba o hacia abajo, y si esta horizontal es hacia la izquierda o la derecha, las direcciones también son relativas al punto de entrada.

****

**Figura 10.** Prueba de definición de punto de entrada..

**Generación del camino**

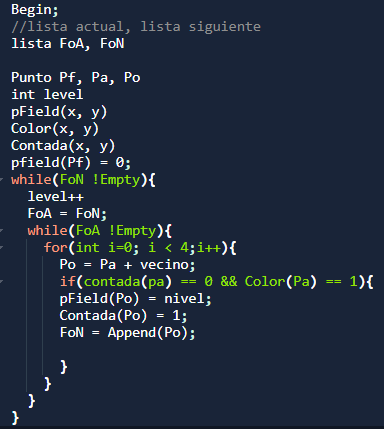
Una vez que se ha definido el radio del robot, se ha binarizado el espacio de trabajo y ensanchando los obstáculos se puede empezar a trabajar en el desarrollo del algoritmo generador de caminos.

Dicho algoritmo se dividirá en 2 partes, la primera será la generación de un **campo de potencial**, en donde el el punto de llegada del robot será el nivel más bajo del campo. La segunda parte el **trazado de la ruta** a seguir, en donde se debe generar una ruta con líneas rectas desde el punto de partida hasta el punto de llegada

**Campo de potencial**

Para general el campo de potencial, se utilizó una variación del algoritmo de de gota de aceite para poder simular la expansión del campo, alrededor del punto de llegada.

El siguiente Pseudocódigo ilustra el algoritmo utilizado



**Figura 11.** Pseudocódigo del algoritmo de gota de aceite modificado

De esta forma, se tienen dos listas, la primera se utiliza para explorar a todos los vecinos actuales, y asignarles el mismo nivel de potencial en la matriz de pfield, mientras que la siguiente almacena los siguientes vecinos que conformarán el siguiente nivel de potencial.

A los obstáculos por default se les ha dado el mayor número de potencial.

Así, se puede obtener el campo de potencial para cualquier cajón de estacionamiento, como se muestra en la siguiente figura



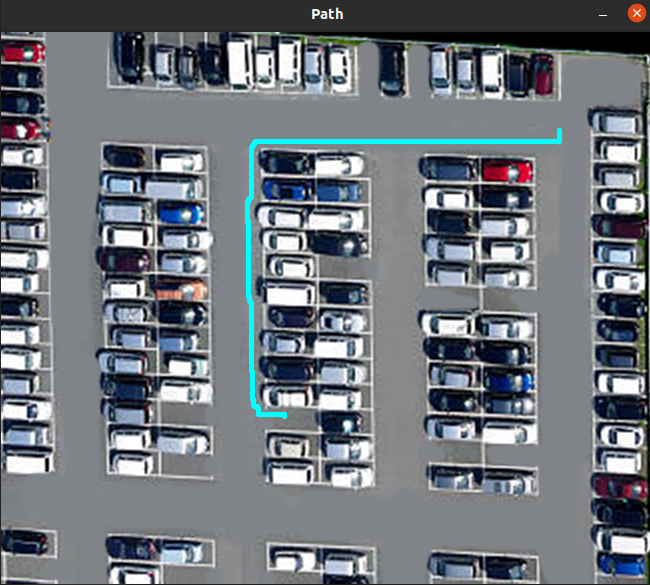
**Figura 12.** Campo de potencial

En la figura anterior se muestra el punto de llegada en rojo, y el campo de potencial tiene su valor más bajo (más oscuro), desde ese punto cada pixel que se aleje tendrá un valor más alto (más claro).

**Trazado de la ruta**

Para esta parte, se desarrolló un algoritmo en el que al darle un punto de inicio, este se moverá a través del campo de potencial hacia el píxel vecino con el valor más bajo. En caso de que se encuentre en un punto en el que dos vecinos tengan el mismo valor, se le da preferencia al vecino que mantenga la dirección anterior de avance, de esta forma se logran trazar caminos de la forma más recta posible.

Continuando con el ejemplo anterior, ahora se traza el camino iniciando desde el cuadrante superior derecho

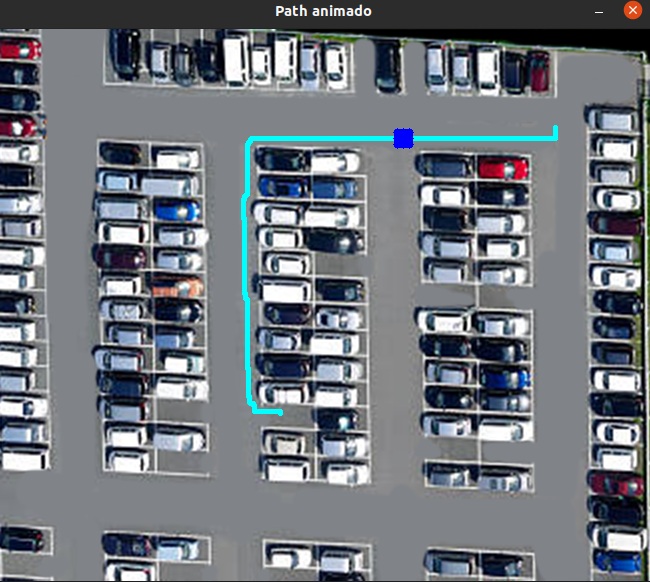


**Figura 13.** Trazado de caminos

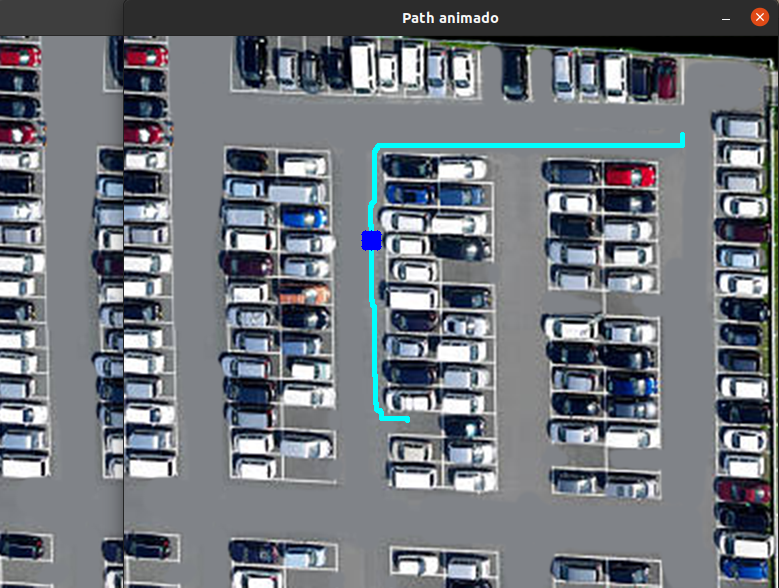
Aunque el cálculo se hace utilizando la matriz de campo de potencial, la ruta se puede trazar directamente sobre la imagen original del estacionamiento, ya que ambas mantienen la misma cantidad de pixeles, columnas y filas.

**Seguimiento virtual del camino**

Para realizar el seguimiento del camino, se utilizó la imagen con el camino trazado y una lista con los puntos seguidos, después se crea una copia temporal de la imagen, y sobre esta se dibuja un marcador en cada iteración de la lista de puntos hasta llegar a la meta. Esto crea una pseudo animación de un objeto moviéndose.

****

**Figura 14.** Animación parte 1

****

**Figura 15.** Animación parte 2

**Conclusiones**

**Melchor Daniel:**

En esta etapa final del proyecto, integramos todos los conceptos y herramientas que vimos y utilizamos en clase durante el semestre. Desde el procesamiento de imagen y filtrado de la etapa 1, a la segmentación, clasificación y reconocimiento de la etapa 2, y finalmente añadiendo lo visto durante las últimas clases de planeación y seguimiento de caminos. Este proyecto resultó ser un desafío, principalmente por la fecha de entrega siendo tan cerca de la de otros proyectos y exámenes, pero gracias a nuestra organización cómo equipo logramos realizar por completo el proyecto.

En esta entrega, mi aportación personal fueron los cambios a las imágenes y un nuevo entrenamiento para corregir nuestros errores de la etapa anterior, además de la función para determinar el punto de entrada del robot utilizando el reconocimiento de imágenes de la etapa 2.

**Alonso ugarte:**

Con esta parte del proyecto, se consolidaron todos los conocimientos obtenidos durante el semestres. Las técnicas de visión computacional como el filtrado, manipulación de imágenes, binarización y reconocimiento de objetos fueron esenciales para desarrollar y probar de manera efectiva un algoritmo de trazado de caminos. Además, el haber tenido el desarrollo técnico y entendimiento teórico nos permite modificarlas y adaptarlas a nuestra aplicación en específico. Aunque para este caso en específico el código fue creado en C++, y se tuvo apoyo de las librerías de opencv, el hecho de conocer los mecanismos internos de cada técnica utilizada nos permitirá utilizar y crear algoritmos en cualquier plataforma.

También, el tener que integrar diferentes partes, tanto de este entregable como de los entregables anteriores fue un rato técnico, con lo cual hemos aprendido la importancia de realizar código de forma modular.

Por último, mi aportación personal a esta parte del proyecto fue el desarrollo del algoritmo para la planeación de caminos, y la animación del objeto por la ruta trazada.

**Yelanetzi Chávez:**

Esta parte del proyecto fue muy útil como recopilación de todos los conocimientos previos. El filtrado visto desde las primeras clases nos ayudó como base para generar imágenes limpias y para posteriormente trabajar en ellas. Las técnicas de segmentación del segundo parcial nos ayudaron a clasificar y segmentar los objetos que presentamos en pantalla. Las investigaciones previas nos ayudaron a entender el funcionamiento correcto de los algoritmos y a entenderlos a profundidad para modificarlos y que sean de apoyo de acuerdo con nuestros objetivos. En este trabajo aprendí la importancia de la investigación ya que necesitábamos conocer las variaciones del algoritmo de gota de aceite y era importante también saber las funciones con las que cuenta OpenCv y que iban a ser de ayuda.

En esta entrega, mi colaboración fue en los puntos 3 y 4 de la rúbrica, que consistía en binarizar la imagen, ensanchar los obstáculos y con un click del mouse encontrar un espacio disponible para crear el camino hacia dicho cajón de estacionamiento. También trabajé en la integración de cada uno de los puntos para optimizar y mejorar la ejecución de nuestro programa.

**Antonio Franco:**

La etapa final fue una forma muy interesante de poder resumir todo el curso, al aplicar todas las técnicas que hemos visto desde el principio. En un inicio, el proceso de captura y filtrado, segmentación, caracterización, clasificación e interpretación me parecía algo difícil de imaginar, pero en esta estapa final me doy cuenta de la importancia de generar soluciones correctas y robustas en cada una de las fases del proceso, porque una ligera variación genera grandes errores y problemas en las siguientes partes. Específicamente hablando de la última etapa, nuestra mayor complejidad fue integrar los códigos de cada integrante del equipo, debido a que la máquina virtual no soportaba tener tantas ventanas abiertas al mismo tiempo y se colapsaba. Al final, tuvimos que destruir ventanas que se iban acumulando al momento de usar el programa para que pudiera correr por completo sin necesidad de cerrar y abrirlo de nuevo En cuanto a mi colaboración en esta etapa final, estuve involucrado en capturar la imagen y editarla usando diferentes técnicas de binarización para dejar solo las calles y los espacios de estacionamiento. También definí cuáles son los obstáculos y el radio del carro.

**Bibliografía:**

Dadhich, A. (2018). *Practical computer vision: Extract insightful information from images using TensorFlow, Keras, and OpenCV*. Birmingham, UK: Packt Publishing

F., V. S., & F., V. S. (2004). Visión por computador. Madrid: Dykinson.

J. Shi and J. Malik. Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(8):888–905, 2000.

X. Ren and J. Malik. Learning a classification model for segmentation. In Proc. ICCV, pages 10–17, 2003

J. Ning, L. Zhang, D. Zhang, and C. Wu. Interactive image segmentation by maximal similarity based region merging. Pattern Recognition, 43(2):445–456, 2010.

**Anexo:**

[Enlace al cálculo de datos.](https://docs.google.com/spreadsheets/d/1i1gyaG0IiqcJlDxcNs_8WBZ1HWBdLO7Omn4h3gK0hbY/edit?usp=sharing)

[Video de demostración.](https://drive.google.com/file/d/1ipjL2bgLeP5VY8lNH3-U00Ndjp4grhL1/view?usp=sharing)

[Repositorio de Github.](https://github.com/MrDaniRamo/Equipo4Vision)