Área Académica Ingeniería en Mecatrónica

Curso: Sistemas de Visión

Memoria Escrita para el Proyecto Final

Profesor:

Juan Luis Crespo Mariño

Estudiantes (Grupo 10):

David Quesada Solís, 2015100428

Emanuel Venegas Mayorga, 2015027049

II Semestre

2021

Índice

Descripción del problema

La idea principal del proyecto es realizar un sistema de interpretación de instrucciones representadas por una serie de símbolos y colores, las cuales se encuentran impresas en tarjetas que serán captadas por una cámara de video. Estas instrucciones influyen en el movimiento de un objeto que se mueve por el circuito cerrado de la Figura 1, el cual cuenta con varias intersecciones y mallas cerradas. Este circuito y el objeto, en este caso representado como un carro, serán programados en su totalidad dentro de un interfaz gráfico [1].

Una consideración importante del problema es que las tarjetas presentan elementos ruidosos, por lo que el sistema de visión debe ser suficientemente robusto para poder reducir o eliminar el ruido, y así realizar una interpretación adecuada de la instrucción que se esté captando [1].

Figura 1. Circuito en el que se traslada el carro. Fuente: Juan Crespo [1]

Metodología

Para el proyecto, se plantean los siguientes pasos a realizar:

- Identificar los bloques funcionales que requiere la solución para el cumplimiento del objetivo del sistema, utilizando aspectos y técnicas relacionados a sistemas de visión.
- Plantear el sistema de visión a utilizar, es decir, aspectos relacionados a la cámara, iluminación y transmisión de video a la aplicación a desarrollar.
- Plantear la apariencia de las tarjetas teniendo en cuenta especificaciones de ruido y elementos ruidosos.
- Determinar los parámetros del sistema y las técnicas necesarios para la extracción e interpretación adecuada de una instrucción por medio de un análisis de las características del entorno físico del sistema y de las tarjetas.
- Definir las variables de respuesta (objetivo) que se desea estudiar así como la serie de valores a poner a prueba para cada variable.
- Definir los factores de influencia que inciden en las variables de respuesta, así como el conjunto de valores a poner a prueba para cada factor.
- Escribir el código relacionado a la extracción e interpretación de instrucciones.
- Escribir el código relacionado a la simulación del movimiento del carro sobre un circuito.
- Construir el sistema de visión, teniendo en cuenta cierta flexibilidad para posibles modificaciones durante las pruebas.
- Fabricar las tarjetas para la validación del sistema dentro de pruebas.
- Poner en marcha el experimento para validar el sistema, apuntando los datos obtenidos de forma ordenada.
- Realizar un análisis de los resultados.

Descripción general de la solución a implementar

Dada la naturaleza del problema, se entiende que la solución se divide en 2 partes fundamentales: un algoritmo que detecte correctamente las instrucciones, y otro que las implemente para controlar un carro simulado dentro de un interfaz gráfico. En el primer programa, llamado “Programa de detección de instrucciones” (Figura 2), en primer lugar se debe captar la imágen de la cámara, después se realiza un preprocesado para eliminar elementos no deseados, luego un módulo de segmentación para extraer las figuras a analizar, y por último un módulo de interpretación que encuentra y compara los patrones que corresponden a los comandos para poder enviarselos a la segunda parte del programa, llamada “Programa de control de carro” (Figura 2). En este, se simula un carro que se traslada dentro de un circuito, de igual forma, programado. El carro se puede controlar, si se desea, alimentando esta parte del programa con los comandos obtenidos del algoritmo anterior, por medio de una decodificación de los comandos en instrucciones más simples que el algoritmo de control del carro puede entender. Ambas partes deben ejecutarse al mismo tiempo de forma paralela para que el proyecto en su totalidad funcione de forma adecuada. En la Figura 2 se pueden lo descrito anteriormente, en forma de diagrama de bloques.

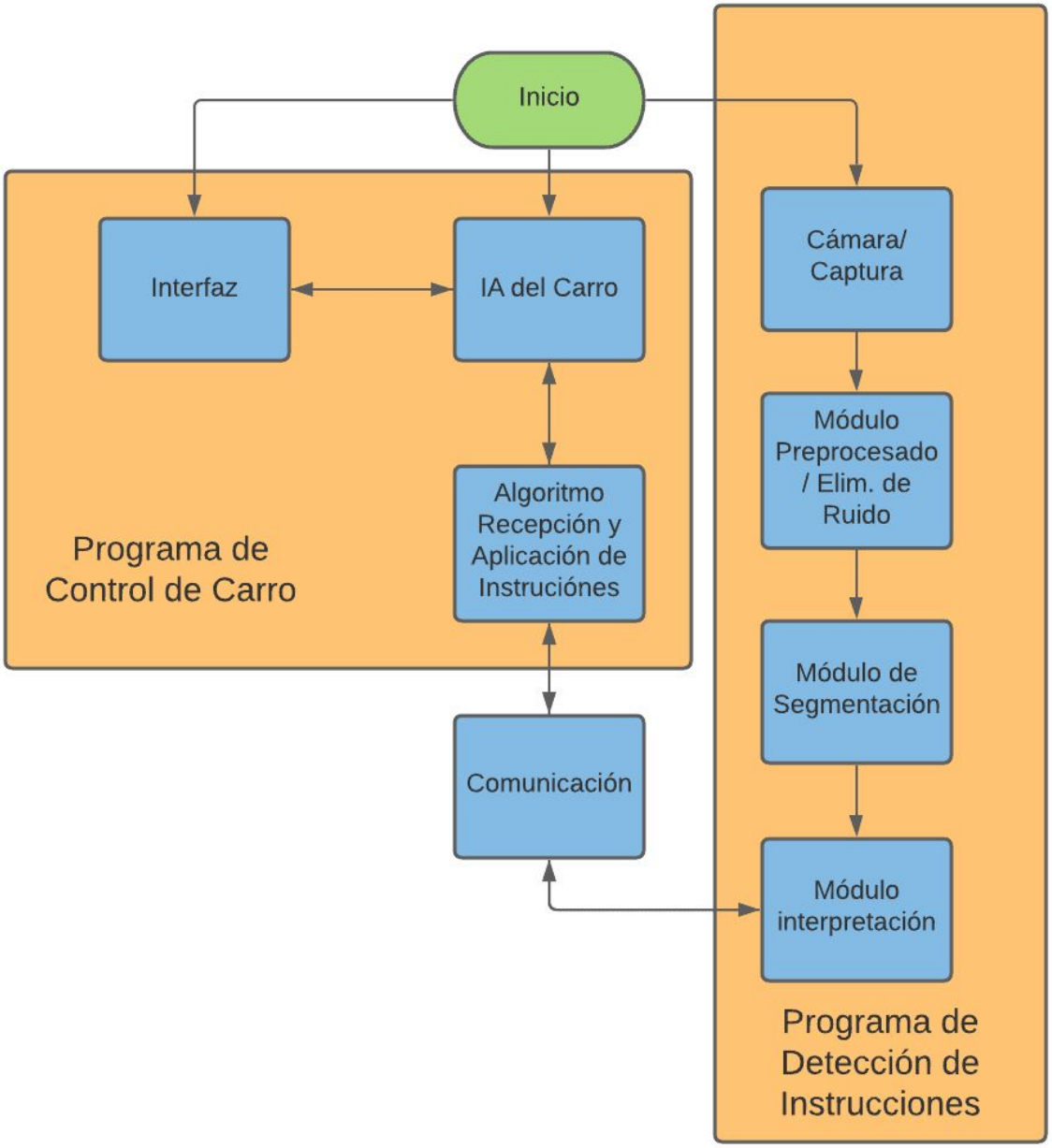


Figura 2. Diagrama de bloques del programa. Fuente: elaboración propia.

Captura de video e iluminación

Esta parte del proyecto es fundamental para una adecuada toma de imagen de la tarjeta, y reduce la carga computacional del algoritmo de identificación del patrón correspondiente [2].

Captura de video

Debido a la complejidad relativamente baja del proyecto respecto a este aspecto, una cámara tipo web es suficiente. Además, una de las computadoras a utilizar durante el proyecto posee una integrada (laptop); esto tiene varias ventajas, incluyendo que es que es fácil posicionarla y que no se requiere configurar para que la computadora la reconozca. Dicha cámara (Rotulada HD WebCam, Hardware ID: USB\VID_0408&PID_A060&MI_00) es de 0.9 MP, pero para reducir el tamaño de la imágen captada, se usará con una resolución de 0.3 MP, con una relación 4:3 (480, 640). Esta resolución anterior es adecuada, ya que el tamaño relativo de las figuras a medir es suficientemente grande.

Escena

La escena puede ser cualquiera cuando no se tiene una tarjeta en el campo visual de la cámara. Sin embargo, las tarjetas toman prioridad una vez entran en el campo visual.

Iluminación

Se utilizará iluminación frontal directa en campo claro, ya que las tarjetas serán impresas en papel, el cual no generará reflexiones. Además, se hará así para garantizar que llegue la energía suficiente al sensor de la cámara.

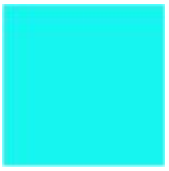
Tarjetas

Por restricción del proyecto, las tarjetas deben ser de 15 x 10 cm2, y deben poseer el comando como un símbolo. En la tarjeta, la figura que representa la instrucción, abarca un máximo del 15%, del espacio de la tarjeta, además contará con figuras similares con el fin intentar confundir al sistema de detección, por lo cual las figuras tendrán entre 1 a 2 características, además de agregar manchas blancas y negras para tener ruido sal y pimienta[1]. Para la creación de las fichas se utilizó una herramienta de edición de imágenes, para crear todas las variantes de las fichas, se crean varias capa que incluyen las figuras de fondo que son posiblemente confundible con la instrucción; instrucciones en diferentes punto de la fichas y con ángulos de rotación distintos; y ruido sal y pimiento para simular imperfecciones de la cámara.

Definición de símbolos

De igual forma, se requiere reconocer patrones específicos que corresponden a las acciones a interpretar por el programa. En la Tabla 1 se aprecian estos símbolos. Cualquier otra configuración de figuras y colores no se debe reconocer por el programa como uno de los comandos disponibles.

Tabla 1. Símbolos a interpretar.

No. Figura	Patrón	Instrucción	Descripción
1		Gira a la derecha y acelera	Cuadrado con color naranja, RGB (245, 200, 23).
2		Gira a la izquierda y acelera	Triángulo con color naranja, RGB (245, 200, 23).
3		Gira a la derecha y desacelera	Cuadrado con color celeste, RGB (23, 245, 240).
4		Gira a la izquierda y desacelera	Triángulo con color celeste, RGB (23, 245, 240).
5		Detención	Signo “+” negro, sólo se reconoce como “Detención” si se encuentra dentro de una de las figuras 1 a 4.
6		Puesta en marcha	Signo “-” negro, sólo se reconoce como “Puesta en marcha” si se encuentra dentro de una de las figuras 1 a 4.

Desarrollo del programa

Programa de detección de instrucciones

Lógica de reconocimiento

Para el implementación del programa de detección, se caracteriza los valores de color y los rangos de insertidumbre. Se inicia con la detección de los colores, se realiza la conversión de formato RGB a HSV, de los colores dando Naranja HSV (48, 90.6, 96.1) y Turquesa (179, 90.6, 96.1); dando una variación de un 10% en el matiz, y 30% en la saturación e valor; estos se les da mayor rango puestos son más afectados por la iluminación del entorno. Con esta caracterización se crean máscaras que separan los colores deseados del resto de la imagen.

Con las figuras del color deseadas separadas de las imágenes; se realiza una operación de búsqueda de bordes; de las figuras restantes; después se cuentan los lados de las figuras cerradas por encontrar los que son triángulos y cuadrados; de otro tipo de figuras que pudieran tener el mismo color. Con las figura y el color detectado se da la primera instrucción al sistema.

El paso final es buscar en la figura seleccionada, la existencia de un signo; se separa de la imagen la región de la figura; y se realiza una mascara que busque el color negro en la sección restante, seguido de una operación de busqueda de contorno; lo que finalmente de exclusivamente los objetos dentro de la figura; se cuentan los lados de las figuras restantes, y se determina si es un signo de resta o suma; se se encuentran cualquiera de los se actualiza una variable para dirrecionar el carro; en caso contrario se mantene la instrucción dada por el paso previo.

Programa de control de carro

Lógica y construcción del circuito

En primer lugar, se tuvo que programar la ruta del carro, para que el mismo carro tenga las coordenadas de todas las intersecciones. Para ello, se crearon 2 clases de Python:

- Clase Point:
    - Es la clase con la que se representan los puntos de intersección de cada una de las rectas del circuito. Por cada instancia de punto, se guarda los puntos conectados a éste por medio de una recta (llamados puntos vecinos o neighbors), así como la dirección (norte, este, sur u oeste) en la que residen tales puntos desde el punto de vista de la instancia original.
- Clase Circuit:
    - Es la clase con la que se representa el circuito completo. Por medio del método addNeighborToLast(), se puede construir, recta por recta, un circuito, agregando un nuevo punto del circuito al punto agregado previamente como un vecino. Además, con el método joinLastTo(), se pueden agregar puntos vecinos ya existentes al último punto agregado, en caso de que aún no exista tal conexión. En la Figura 3 se puede ver el segmento del código en el que se construye el circuito de la Figura 1. Además, en la Figura 4, se puede ver la numeración de los puntos.

```
circuit = Circuit()
circuit.addNeighborToLast(3, points[3])
circuit.addNeighborToLast(9, points[9])
circuit.addNeighborToLast(8, points[8])
circuit.addNeighborToLast(2, points[2])
circuit.joinLastTo(3)
circuit.addNeighborToLast(0, points[0])
circuit.addNeighborToLast(1, points[1])
circuit.addNeighborToLast(5, points[5])
circuit.addNeighborToLast(4, points[4])
circuit.addNeighborToLast(15, points[15])
circuit.addNeighborToLast(16, points[16])
circuit.addNeighborToLast(14, points[14])
circuit.addNeighborToLast(13, points[13])
circuit.addNeighborToLast(11, points[11])
circuit.addNeighborToLast(12, points[12])
circuit.addNeighborToLast(7, points[7])
circuit.addNeighborToLast(6, points[6])
circuit.joinLastTo(11)
circuit.addNeighborToLast(10, points[10])
circuit.joinLastTo(5)
circuit.joinLastTo(6)
```

Figura 3. Generación en el programa del circuito de la Figura 1. Fuente: elaboración propia.

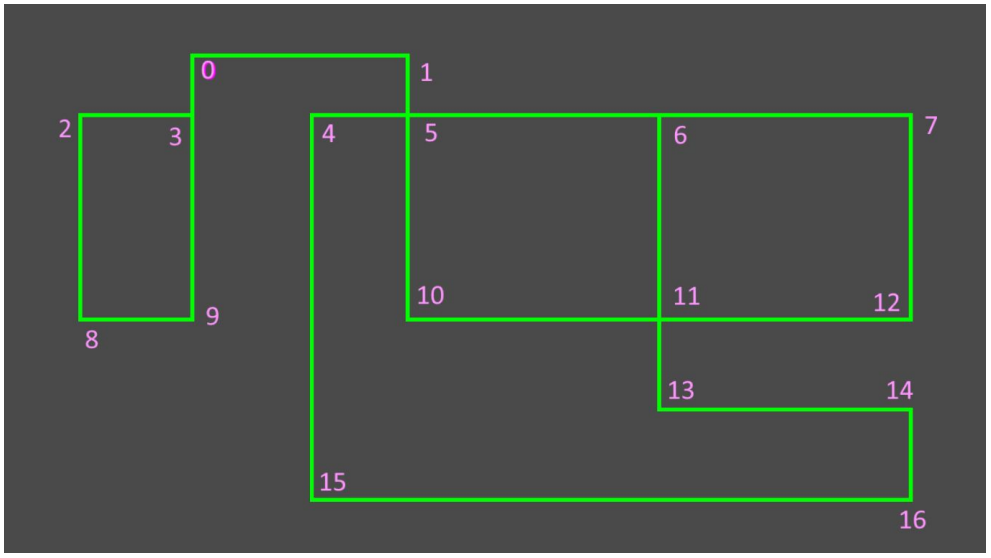


Figura 4. Numeración de los puntos del circuito de la Figura 1. Fuente: elaboración propia.

Lógica del carro

También se creó la clase Car, cuya lógica permite recibir comandos y actúa acorde dentro del circuito por medio de cálculos que utilizan las coordenadas de las instancias de Point. Entonces, el carro espera hasta la siguiente intersección (punto instancia de Point con 3 o más vecinos) para ejecutar el último comando leído de las tarjetas. En el momento que los comandos son ejecutados, reinicia el comando y espera al siguiente comando. Si la instrucción es “detención”, se detiene en la intersección, espera 10 segundos, y sólo reanuda su movimiento si la siguiente instrucción es “puesta en marcha”. Si la última instrucción es alguno de los otros comandos, las ejecuta según sea la instrucción. Por último, se programó un movimiento que no requiere comando, en caso de que no se haya leído ninguna tarjeta previo a una intersección.

Interfaz gráfico

El interfaz gráfico se creó utilizando el módulo Pygame, el cual se utiliza para crear juegos programados con Python, pero funciona bien para efectos de este proyecto. Como se aprecia en la Figura 5, las líneas gruesas negras son el circuito, el punto rojo es el carro, y en la esquina superior derecha se muestran los siguientes valores:

- Sig: siguiente comando.
- FPS: frecuencia de actualización de la imágen mostrada en el interfaz.
- Veloc: la velocidad del carro medida en pixeles por segundo.
- Reloj: reloj que toma el tiempo de 10 segundos cuando se activa el comando “detención”

También, en la esquina inferior izquierda, se incluyó la captura de imagen de la cámara.

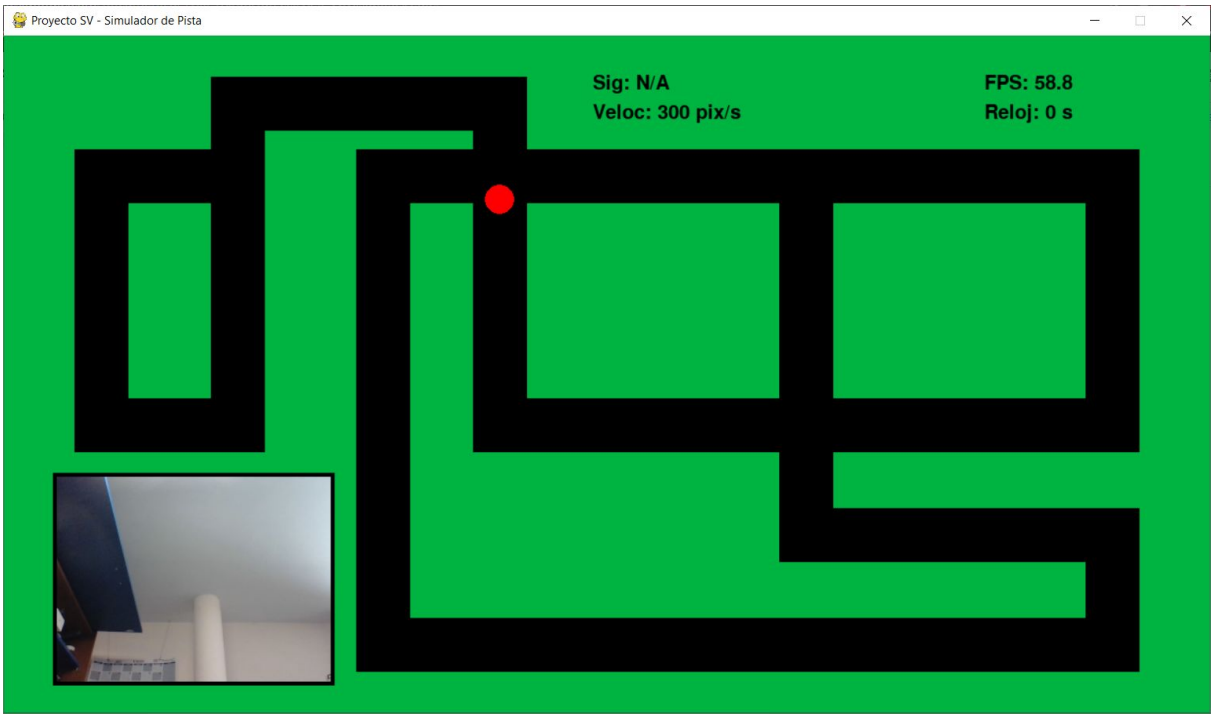


Figura 5. Interfaz gráfico del programa. Fuente: elaboración propia.

Integración

La integración de ambas partes del código se logró por medio del uso de hilos o threads, y una clase memoria. Los threads, del módulo Threading, permiten correr varias funciones de forma paralela, por lo que se corrió la función del interfaz main(), en conjunto con el algoritmo de reconocimiento de instrucciones patternRecognition(). Por otro lado, se creó una nueva clase, llamada ProgramStatus, cuyas instancias se almacena en memoria RAM, y así se puede acceder si se cuenta con la instancia. Por lo tanto, una sóla instancia de ProgramStatus se crea y se comparte a todas las funciones activas, para que tales funciones puedan escribir y leer información del estado del programa sin dificultad alguna. Así se cumple el módulo de comunicación que se muestra en la Figura 2.

Características y diseño de prueba

En esta parte del documento se utilizará parte de la teoría encontrada en [3] para formular pruebas formales. Por lo tanto, primero se definen los objetivos de las pruebas, luego las variables de respuesta y factores de influencia, junto con sus valores para pruebas, y luego se redacta el diseño de los experimentos.

Objetivo de las pruebas

- Validar la funcionalidad y el desempeño del programa de interpretación de instrucciones.
- Validar el sistema de recepción y aplicación de instrucciones al carro.
- Identificar características del subsistema de visión que se requieran modificar para mejorar el desempeño del sistema, de haber.

Factores de influencia y variables de respuesta

Como se requiere validar el sistema entero, los únicos aspectos que se podrían modificar son las características y posición de las tarjetas. Aún así, también se consideró importante agregar un aspecto de iluminación de las tarjetas, el cual es también una característica de la tarjeta, pero es controlada principalmente por la posición relativa de la fuente de luz. Dentro de los parámetros que se pueden modificar en las tarjetas se tienen:

- **Instrucción solicitada:** instrucción específica a transmitir.
- **Tamaño del símbolo:** área en porcentaje de la tarjeta ocupada por el símbolo. Esta no puede ser mayor al 15%.
- **Magnitud del ruido sal y pimienta:** porcentaje de píxeles que contienen un brillo alto y bajo.
- **Cantidad de elementos ruidosos:** cantidad de elementos similares a los símbolos que representan las instrucciones, pero que no cumplen con la definición de un símbolo válido. Estos poseen características aleatorias pero similares al del símbolo de instrucción, y siempre aparecen “debajo” del patrón correcto si se diera el caso.

- **Posición del símbolo en la tarjeta:** coordenadas (x, y) del centro de un círculo circunscrito al símbolo, respecto a la esquina superior izquierda de la tarjeta. Este factor se mide en cm para ambos valores.
- **Orientación del símbolo en la tarjeta:** ángulo respecto a la orientación de cada instrucción definida previamente. Este factor se mide en grados.

Respecto a las variables de respuesta se tienen:

- **Interpretación adecuada de la instrucción:** equivalencia entre la instrucción alimentada al sistema y la interpretada.
- **Actuación adecuada por parte del carro:** concordancia entre la instrucción alimentada al sistema y el movimiento del carro.

Rangos de prueba y tratamientos

Dado que la cantidad de factores es amplia, se dividirá esta etapa en 2 partes. Las primeras pruebas son para comprobar el rendimiento del sistema, utilizando varios valores de los factores que se consideran que pueden confundir más al sistema de interpretación. Aún se requieren variar muchos factores, por lo que se considera bloquear el factor de instrucción solicitada y el tamaño del símbolo, y se utilizarían un solo valor para este caso en ambos factores (Valores correspondientes: “Gira a la derecha y acelera” y 10%). Se considera además combinar los factores de posición y ángulo para poder aplicar varios casos de ambos factores, sin aumentar el número de corridas. Por lo tanto, los tratamientos a poner a prueba se pueden ver en la Tabla 2. Se realizará una sola repetición por tratamiento, lo que resulta en 20 corridas.

Tabla 2. Matriz de diseño de las primeras pruebas a realizar.

No. Tr.	A: Magnitud ruido	B: Cant. elementos ruidosos	C1: Posición	C2: Angulo (°)
1	3	2	(3.0, 2.7)	0
2	3	2	(12.0, 4.1)	72
3	3	2	(12.1, 7.0)	144
4	3	2	(4.4, 5.7)	216
5	3	2	(7.8, 4.6)	288
6	3	4	(3.0, 2.7)	0
7	3	4	(12.0, 4.1)	72
8	3	4	(12.1, 7.0)	144
9	3	4	(4.4, 5.7)	216
10	3	4	(7.8, 4.6)	288
11	10	2	(3.0, 2.7)	0
12	10	2	(12.0, 4.1)	72
13	10	2	(12.1, 7.0)	144
14	10	2	(4.4, 5.7)	216
15	10	2	(7.8, 4.6)	288
16	10	4	(3.0, 2.7)	0
17	10	4	(12.0, 4.1)	72
18	10	4	(12.1, 7.0)	144
19	10	4	(4.4, 5.7)	216
20	10	4	(7.8, 4.6)	288

En la segunda etapa de experimentación, se realizarán pruebas que están más relacionadas con la funcionalidad del sistema, es decir, si interpreta bien los símbolos. En esta etapa, se mantendrá en bloques los factores de las pruebas anteriores (Magnitud ruido: 5%, cant. elementos ruidosos: 4, posición: (9.5 cm, 6.0 cm), ángulo: 120°), y se varían los factores de instrucción solicitada y tamaño del símbolo. Se deben probar las 6 instrucciones, por lo que el otro factor debe tener valores reducidos. Entonces, se realizará una corrida por tratamiento, lo que resulta en 12 corridas en total. Los tratamientos a poner a prueba se pueden ver en la Tabla 3.

Tabla 3. Matriz de diseño de las segundas pruebas a realizar.

No. Tr.	A: No. Instrucción	B: Tamaño símbolo (%)
1	1	12
2	1	6
3	2	12
4	2	6
5	3	12
6	3	6
7	4	12
8	4	6
9	5	12
10	5	6
11	6	12
12	6	6

Resultados y Análisis

Tabla 4. Resultados de las primeras pruebas

No. Tr.	Interpretación adecuada de la instrucción	Actuación adecuada por parte del carro
1	Si	Si
2	Si	Si
3	Si	Si
4	Si	Si
5	Si	Si
6	No	No
7	Si	Si
8	Si	Si
9	Si	Si
10	Si	Si
11	Si	Si
12	Si	Si
13	Si	Si
14	Si	Si
15	Si	Si
16	Si	Si
17	Si	Si
18	Si	Si
19	Si	Si
20	Si	Si

Tabla 5. Resultados de las segundas pruebas

No. Tr.	Interpretación adecuada de la instrucción	Actuación adecuada por parte del carro
1	Si	Si
2	No	No
3	Si	Si
4	Si	Si
5	Si	Si
6	Si	Si
7	No	No
8	Si	Si
9	Si	Si
10	Si	Si
11	No	No
12	No	No

La primera prueba se obtiene 19 interpretaciones correctas de 20 posible; esta da una éxito del 95 por ciento de reconocimiento; en todas las ocasiones el carro siguió la interpretación del sistema de la tarjeta; durante el las pruebas se dan falso positivos con la señal de puesta en marcha; esta se debe a ruido en el ambiente que el sistema confunde con la señal. El sistema no presenta sensibilidad a ruido en la tarjeta; pero si a ruido del ambiente.

En la segunda prueba se obtiene 8 interpretaciones correctas de 12 posible; lo cual da un éxito del 66 por ciento; en las 4 ocasiones de error 3 de ellas son cuando el símbolo tiene una dimensión; por lo se puede concluir que el sistema de reconocimiento no es efectivo con figuras de 6% de tamaño o menor.

Conclusiones

- Se logra desarrollar un sistema de detección, con una tasa aceptable de éxito siempre y cuando el tamaño del símbolo sea al menos de un 10%
- El sistema muestra sensibilidad a cambios del ambiente; como iluminación u objetos con extraños con tonalidades similares a las de las instrucciones
- La interfaz del sistema es desarrollada con éxito; y sigue a la perfección las instrucciones dadas por el algoritmo de reconocimiento

Referencias bibliográficas

[1] J. Crespo, *"TAREA EVALUABLE PROYECTO DE CURSO: SIMULADOR DE GUIADO AUTÓNOMO DE UN VEHÍCULO"*, sin publicar.

[2] J. Crespo, *"Tema 2: CONSIDERACIONES SOBRE LUZ E ILUMINACIÓN"*, sin publicar.

[3] H. Gutiérrez Pulido and R. de la Vara Salazar, *Análisis y diseño de experimentos*, 3ra ed. México: McGraw-Hill, 2012, p. 2-11.