Navegación por Procesamiento de Imágenes Usando OpenCV

Gallardo García Hugo y Juárez Parga José Miguel

Instituto Tecnológico y de Estudios Superiores Monterrey Campus San Luís Potosí
Av. Eugenio Garza Sada 300, Lomas del Tecnológico, San Luis Potosí, S.L.P. México

Resumen

Este artículo contiene el desarrollo de un vehículo terrestre autónomo utilizando el procesamiento de imágenes como medio de navegación. Se mostrará los datos y la información de procesamiento de imágenes así como la descripción del hardware, la lógica de navegación y los dispositivos utilizados en el proyecto.

Palabras clave:

Arduino: plataforma de electronic abierta para creación de prototipos.

OpenCV: es una librería gratuita de visión artificial y procesamiento de imágenes originalmente desarrollada por Intel.

RGB: modelo de color basado en las componentes roja, verde y azul de la luz.

YUV: modelo de color utilizado principalmente para señales de video en el que los colores se especifican de acuerdo con su luminosidad, su tono y la saturación.

Microsoft Visual Studio: Entorno de desarrollo integrado de Microsoft.

Detección de bordes de Canny: Técnica desarrollada por John Canny F. en 1986 que utiliza un algoritmo de múltiples etapas para detectar una amplia gama de bordes en imágenes.

Processing: es un lenguaje de programación de código abierto construido para el arte electrónico y las comunidades de diseño visual, como Arduino.

1. Introducción

Con el uso de la biblioteca OpenCV se desarrolló un programa en Microsoft Visual Studio que es el encargado de realizar las tareas de reconocimiento de imagen, imagen filtrada y la generación de trayectorias.

La imagen que vamos a reconocer y con la que vamos a trabajar es un escenario con círculos rojos que representan los obstáculos y un coche de radio control que es nuestro vehículo autónomo. Una webcam conectada al puerto USB de la computadora es el sensor que le da información al software para realizar el control del vehículo a través del escenario evadiendo obstáculos.

Los obstáculos que el coche tiene que evitar se detectan con el uso de funciones de la librería OpenCV usando la información de vídeo de la webcam. El programa genera una trayectoria, evitando todos los objetos y se envía una señal al hardware de Ardunio a través del puerto USB. Esta señal se interpreta y define el control realizado por microcontrolador de Atmel TMEGA 328-P. Las señales se envían al vehículo con un emisor de radio frecuencia.

2. Diagrama de bloques principal

La figura 1 muestra las etapas del proyecto de manera general. Se sabe que un diagrama de bloques es esencial para desarrollar un desarrollo de programación ya que te permite visualizar los objetivos y dividir el mismo programa en etapas.

El programa se dividió en dos partes. Una es el código de procesamiento de imágenes en Visual Studio utilizando lenguaje C++ y las funciones de la librería OpenCV. La otra parte es el código de control en Arduino Processing usando lenguaje C++ con las librerías de Arduino que facilita la programación y el control del vehículo.

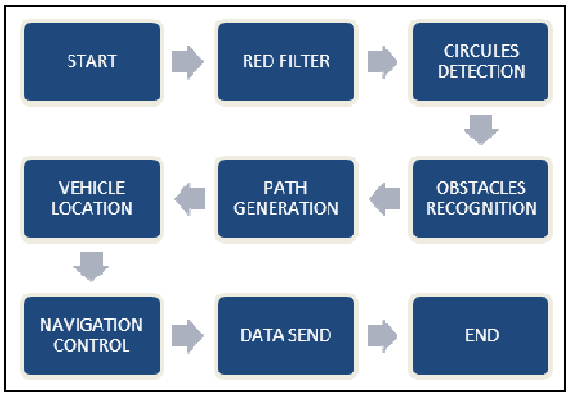

Fig. 1. Diagrama de bloques del proyecto

3. Procesamiento de imágenes

Básicamente el procesamiento de imágenes se basa en capturar una imagen de la webcam y después modificarla utilizando funciones de la librería de OpenCV. Para este proyecto se usaron las funciones cvCapturefromCAM y cvCapture para obtener las imágenes de nuestra webcam.

Un filtrado en la imagen es una técnica muy útil cuando se trabaja con captura de imágenes desde una webcam. El propósito principal de este proceso es separar los colores y eliminar colores y/o secciones de la imagen que son consideradas inservibles. Los filtros más usados son los umbrales de escala de grises, RGB, YUV y otros. Para este proyecto se utilizó el modelo YUV para facilitar el filtrado. **En procesamiento de imágenes de video los canales de brillo, intensidad y saturación saturación (YUV) son más importantes que los canales RGB**, sin embargo, los canales RGB son necesarios para calcular los canales YUV. El modelo YUV te facilita filtrar la información que te interesa y es necesario considerar los cambios de luz en el programa al momento de realizar las capturas de la webcam. A continuación se muestra las ecuaciones para obtener los canales YUV:

$$Y = (0.299 \cdot R) + (0.587 \cdot G) + (0.114 \cdot B) \quad (1)$$
$$U = 128 - (0.169 \cdot R) - (0.331 \cdot G) + (0.500 \cdot B) \quad (2)$$
$$V = 128 + (0.500 \cdot R) - (0.418 \cdot G) - (0.081 \cdot B) \quad (3)$$

Para realizar el procesamiento de imagenes de video se necesitan accede a la información de los canales RGB para después calcular con canales YUV utilizando las ecuaciones 1,2 y 3.

Fig. 2. Vista previa del escenario.

Como se observa en la figura dos, la webcam realiza una captura del escenario, el vehículo y los obstáculos. El segundo paso es el filtrado. Para realizar esto se convierte la imagen a escala de grises como se muestra en la figura tres.

Fig. 3. Imagen en escala de grises.

Se necesita la imagen es escala de grises para poder aplicar el detector de bordes de Canny con una función de OpenCV llamada cvHoughCircles. Esta función nos ayuda a detectar geometrías circulares en la imagen capturada.

4. Detección de Objetos

El programa, como se mencionó anteriormente, tiene una función que reconoce cualquier forma circular de la imagen y después mediante otras funciones se les dibuja un círculo virtual alrededor de cada forma. Sin embargo nuestro vehículo tiene que ser identificado, por

eso se le instaló un círculo verde. El círculo verde nos es útil para conocer la ubicación del vehículo calculando las coordenadas del centro del círculo.

El algoritmo de Canny identifica cualquier forma circular por eso se utilizó un filtro de color que consiste en acceder a los canales YUV de la imagen con el objetivo de implementar un doble condicional que muestre los obstáculos en color rojo. De la misma manera se utilizó el doble condicional para identificar la ubicación del vehículo.

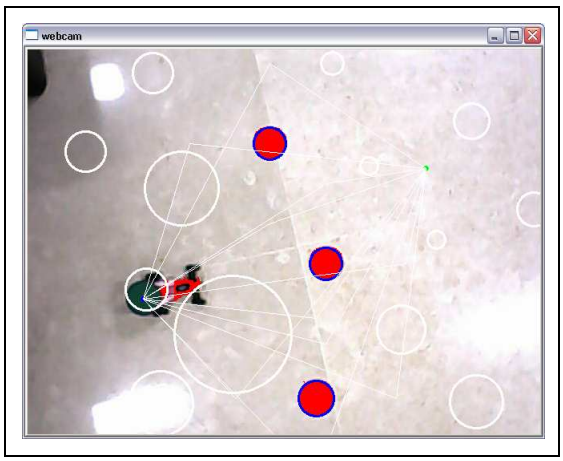

Fig. 4. Detección de obstáculos.

Con el filtro, se puede ver todo en un solo color con diferente intensidad, que es lo que llamamos el umbral de intensidad, podemos reconocer el color rojo, o cualquier otro color que se desee en base a sus componentes YUV. En la figura 4, se observa un brillo o una mayor intensidad en los círculos rojos, así mismo el cuerpo del vehículo es de color rojo, por lo tanto, vemos que es muy brillante. Ocurre esto, porque se programó para pintar cualquier intensidad de color rojo a una intensidad totalmente rojo. (R = 255, G = 0, B = 0) Solamente para mostrar al usuario que el proceso del programa está en orden.

5. Generación de Trayectorias

Una vez que se detectaron los obstáculos, se utiliza una función de OpenCV para crear puntos virtuales que definen el punto de partida y el punto final o de meta. Se crea una trayectoria por el programa para unir los puntos.

La lógica de las trayectorias funciona creando líneas desde el punto de partida hasta cuatro puntos diferentes alrededor de cada círculo que representa el obstáculo. Todos los puntos mencionados se unirán usando la función cvLine. Las líneas que evaden los obstáculos serán las trayectorias posibles. Se elige el camino más corto como trayectoria principal del vehículo.

En la figura cinco se observa una línea verde que conecta el punto de partida y el punto de meta. Esta línea es generada por un método numérico que uno los dos puntos sin tocar los círculos que representan los obstáculos. Las líneas, puntos, círculos, etc. solamente representan una ayuda visual para el usuario. La información de los puntos, círculos y líneas son guardados en variables para realizar el cálculo matemático.

La lógica del programa es calcular y elegir un punto medio cercano que no interfiera con un obstáculo para interpolar con el punto inicial y final. Se desarrolló un algoritmo usando el polinomio de LaGrange grado 3 para generar las trayectorias.

Las siguientes ecuaciones muestran la definición para un polinomio de LaGrange de grado tres.

$$L0 = ((x-x1)/(x0-x1))*((x-x2)/(x0-x2)) \quad (4)$$

$$L1 = ((x-x0)/(x1-x0))*((x-x2)/(x1-x2)) \quad (5)$$

$$L2 = ((x-x0)/(x2-x0))*((x-x1)/(x2-x1)) \quad (6)$$

$$y = (L0*y0) + (L1*y1) + (L2*y2) \quad (7)$$

El cálculo de la ecuación 7 define la trayectoria para nuestro vehículo a través del escenario. x1, x2 y x3 de la ecuación son el punto inicial, el punto medio de interpolación y el punto final de la trayectoria. El punto inicial y final son establecidos con el botón izquierdo y derecho del mouse usando la función de OpenCV cvSetMouseCallback, que crea una interrupción en el programa a causa de los eventos en los botones del mouse.

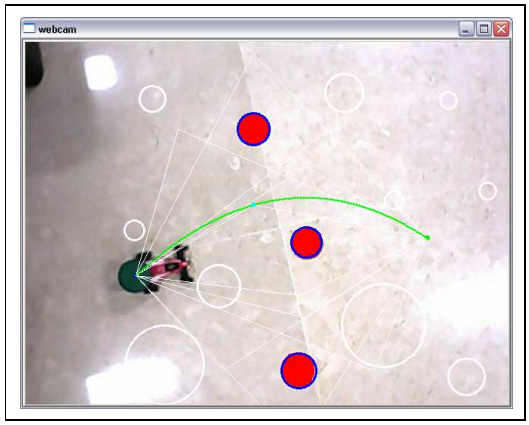

Fig. 5. Generación de trayectoria

6. Navegación

Después de la generación de trayectorias se concluye que la línea verde es una trayectoria segura que el vehículo tendrá que seguir. Se realizó otro algoritmo que hace que el vehículo se mantenga en el camino correcto.

La computadora tiene la función de checar la posición del vehículo solamente reconociendo de cual lado de la línea está. El programa entra en un ciclo continuo hasta que el vehículo llegué al objetivo final.

La lógica de esta sección del programa es calcular el centro del punto verde (ubicación del vehículo) y evaluar las coordenadas del eje "x" en la función de la trayectoria con el objetivo de conocer o bien controlar la dirección del vehículo (derecha o izquierda). Para realizar la decisión es suficiente con comparar la coordenada "y" del punto del vehículo con la coordenada y de la función de trayectoria.

Por ejemplo, si $y_{car} > y_{path}$ gira las llantas a la derecha sino gíralas a la izquierda, repite la evaluación hasta que $x_{car} \geq x_{final\ point}$.

La figura 6 muestra la detección constante de la ubicación del vehículo.

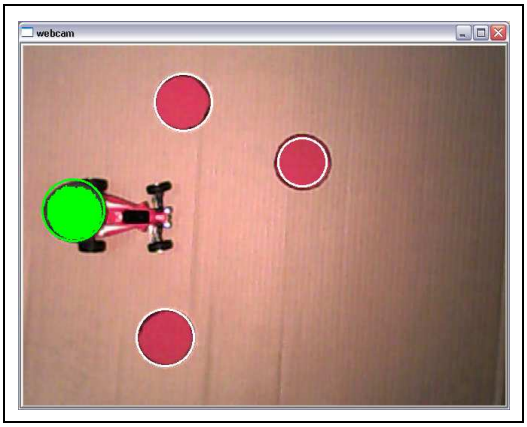

Fig. 6. Ubicación del vehículo

Como se mencionó anteriormente, las señales de avanzar, girar a la derecha, girar a la izquierda y parar son eviadas al arduino usando comunicación serial a 9600 baudios. El Arduino lee constantemente las señales del puerto, las interpreta y manda las señales de control al vehículo.

7. Interfaz de la PC

El hardware de Arduino está conecetado a la PC por el Puerto USB. Este hardware tiene un microcontrolador ATMEGA 328-P y tiene la tarea de controlar el vehículo modificando la dirección y velocidad del vehículo.

El emisor de radiofrecuencia esta conectado al Puerto de salida del arduino; este dispositivo mantiene la comunicación con el recibidor del vehículo. Para construir el dispositivo de comunicación con la PC, se desarmó un emisor de un radio control básico y se adaptó al Arduino.

La lógica del control es sencilla. Se lee constantemente la información que viene de la PC a través del cable USB y selecciona la combinación de señales emitidas al vehículo.

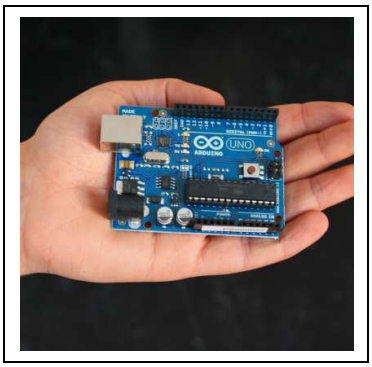

Fig. 7. Plataforma de prototipos Arduino Uno.

8. Vehículo autónomo

El vehículo es un coche de radio control alimentado por una batería recargable LiPo de 3.7V (una celda). Contiene un receptor de 49MHz

Fig. 8. Vehículo Seleccionado

9. Escenario

El escenario es un área rectangular de 1.20m x 1m aproximadamente y depende de la altura donde se posiciona la cámara. Contiene tres obstáculos color rojo con forma circular y con un radio aproximado de 5cm. El problema principal del escenario es el color y el material ya que muchos objetos afectan a al reflejo de la luz.

Para el proyecto se usó una webcam Genius Eye 312. Es una cámara de baja resolución y nos ayudó a hacer un procesado de imágenes más eficiente.

Fig. 9. Webcam seleccionada

Debido a la superficie, se ocasionó un alto brillo o bajo brillo de las imágenes. Por ejemplo, cuando se usó sólo el suelo sin protección o en la superficie, se pudo notar que había manchas blancas debido a los reflejos de la luz sobre el piso.

Se usó una lámpará de escritorio sujetada a una mesa y/o escritorio para iluminar el escenario y variar las alturas de la cámara.

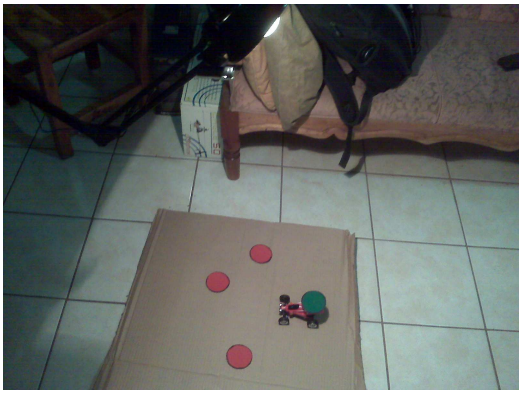

Fig. 10. Fotografía del escenario

10. Resultados

Después de un largo period de pruebas, errores y correcciones de programación se logro integrar las partes del proyecto en un código de 1100 líneas aproximadamente. Se detectaron áreas de mejora como perfeccionar la generación de trayectorias o un sistema de control más complejo. Sin embargo, se satisficieron los objetivos del proyecto al desarrollar un robot móvil autónomo que genera y sigue una trayectoria evadiendo obstáculos rojos usando el procesamiento de imágenes y video.

11. Conclusiones

Los robots móviles terrestres son los elementos más básicos para empezar a desarrollar vehículos autónomos. El dominar las técnicas para control estos robots nos guiará a desarrollar dispositivos más complejos como robots humanoides y vehículos aéreos.

Se aprendió y comprendió que al usar visión procesamiento de imágenes se puede reducir la carga de los robots desarrollados ya que la visión es el sensor más poderoso que te brinda más información que cualquier otro sensor por sí mismo.

Al realizar este proyecto se comprendieron muchos de los problemas al realizar un robot autónomo, como es la toma de decisiones. Sin embargo, este proyecto sirvió para que en el futuro desarrollemos sistemas más robustos en términos de control, procesamiento de imágenes, y vehículos más complejos.

12. Agradecimientos

Nos gustaría agradecer a el Dr. Rolando Cruz, nuestro profesor de robótica, por transmitirnos su conocimiento y guiarnos para el desarrollo de este proyecto. Sin su ayuda el desarrollo de proyectos como este no sería posible.

Referencias

[1] Arduino. (2010). Using Microsoft Visual Studio 2008 to edit, compile and run Arduino code. Recuperado el Febrero de 2010, de Arduino Playground: http://arduino.cc/playground/Code/VisualStudio

[2] Balsero, N., Botero, D., & Zuluaga, J. P. (2005). Reconocimiento e interpretación de gestos manuales por medio de video. Informe final del trabajo de

grado presentado para optar al título de Ingeniero Electrónico . Bogotá, Colombia: Pontificia Universidad Javeriana.

[3] Canny, J. (1986). A Computational Approach to Edge Detection. IEEE.

[4] Cruz, S. R. (Septiembre de 2009). Vision-Based Robot Systems. Partial filfillment of the requirements for the degree of Doctor of Philosophy in Engineering . Osaka, Japón: The Graduate School of Engineering Science Osaka University.

[5] Experimental Robotics. (17 de Junio de 2008). OpenCV Guide. Recuperado el Marzo de 2011, de MediaWiki: http://cgi.cse.unsw.edu.au/~cs4411/wiki/index.php?title=OpenCV_Guide

[6] Intel Corporation. (2001). Reference Manual. Open Source Computer Vision Library . U.S.A.: Intel.

[7] Ramos, E., Morales, R., & Silva, R. (2010). Modelado, Simulación y Construcción de un robot móvil de ruedas tipo diferencial. Ciudad de México: CIDETEC-IPN, Departamento de Posgrado, Área de Mecatrónica.

[8] Russ, J. (2007). The Image Processing Handbook. Raleigh: Taylor & Francis.

[9] Shirriff, K. (25 de Julio de 2009). Secrets of Arduino PWM. Recuperado el Febrero de 2011, de Arduino: http://arduino.cc/en/Tutorial/SecretsOfArduinoPWM

[10] Siegwarth, R., & Nourbakhsh, I. (2004). Introduction to Autonomus Mobile Robots. Cambridge: The MIT Press.

[11] Visual Micro. (Mayo de 2010). Arduino Visual Studio. Recuperado el Marzo de 2011, de Visual Micro making arduino easier: http://www.visualmicro.com/

[12] Wikipedia, The Free Enciclopedia. (23 de Abril de 2011). OpenCV. Recuperado el 26 de Abril de 2011, de Wikipedia: http://en.wikipedia.org/wiki/OpenCV