UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Seguimiento de líneas basado en openCV para AGVs

Presentado por Antonio de los Mozos Alonso
en Universidad de Burgos — 23 de junio
de 2018
Tutor: Jesús Enrique Sierra García

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Jesús Enrique Sierra García, profesor del departamento de Ingeniería Civil, área de lenguajes y sistemas informáticos.

Expone:

Que el alumno D. Antonio de los Mozos Alonso, con DNI 71705119C, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Seguimiento de líneas basado en openCV para AGVs.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 23 de junio de 2018

Vº. Bº. del Tutor:

D. Jesús Enrique Sierra García

Resumen

Los AGV, vehículos de guiado autónomo, son ampliamente utilizados en la industria para transportar cargas. Actualmente estos AGV se guían mediante diferentes métodos, uno de ellos el guiado óptico mediante lineas.

La implementación actual del guiado óptico se basa en una cámara muy cercana y paralela al suelo, que permite detectar si el vehículo está encima de la línea guía o no.

Esta implementación no permite ver en perspectiva lo que el AGV tiene por delante, simplemente detecta si nos salimos de la línea.

En este proyecto veremos diferentes métodos para poder ver en perspectiva lo que el vehículo tiene por delante, y guiarlo de una forma más efectiva.

Descriptores

AGV, vehículo de guiado autónomo, binarización de imágenes, detección de líneas, distorsión por perspectiva, OpenCV, Python.

Abstract

AGV, automated guided vehicle, are widely used in the industry in order to transport loads. Currently these AGV are guided by different methods, one of them the optical track guidance.

The current implementation of the optical track guidance is based on a camera very close and parallel to the ground, which allows detecting if the vehicle is above the guideline or not.

This implementation does not allow to see in perspective what the AGV has in front of it, it simply detects if we get out of line.

In this project we will see different methods to see in perspective what the vehicle has in front, and guide it in a more effective way.

Keywords

AGV, automated guided vehicle, image binarization, line detection, perspective distortion, OpenCV, Python.

Índice general

Índice de figuras

Índice de tablas

Introducción

Los AGV, vehículos de guiado autónomo, actualmente son ampliamente utilizados en el mundo industrial. Su función principal es la de llevar algún tipo de carga de una zona a otra dentro de un recinto. Para ello utilizan distintos métodos de guiado, desde detección de pulsos o bandas electromagnéticas, hasta guiado óptico con cámaras.

Los métodos actuales de guiado óptico se basan en colocar una cámara muy cerca del suelo, en una posición paralela a este, de tal forma que puedan ver una "guía", una linea de algún tono que contraste con el suelo, para guiarse. El problema de esta solución, es que la imagen que la cámara es capaz de ver es un pequeño fragmento de la línea, con el que únicamente es capaz de detectar si está o no sobre la línea, permitiendo corregir la trayectoria cuando se detecta que el AGV se desvía, cuando "pierde la línea".

Esta solución, se puede mejorar si aplicáramos algunos de los métodos que se están usando para el guiado de vehículos autónomos en carretera, donde no solo se detecta si el coche está dentro de los margenes del carril, sino que además son capaces de ver en perspectiva lo que tienen delante. Esto permite, además de guiar el vehículo, estimar la trayectoria que tiene que seguir en los siguientes instantes, permitiendo realizar ajustes más precisos tanto en la velocidad como en la dirección. Se podría decir que son capaces de ver su futuro, el sitio por el que tienen que pasar a continuación.

En este proyecto, se buscará implementar una funcionalidad que permita realizar este tipo de guiado en perspectiva, en el ámbito de la robótica industrial.

Objetivos del proyecto

2.1. Objetivos generales

Desarrollar un software capaz de interpretar una imagen, para generar un conjunto de instrucciones.

- Distinguir en esa imagen una línea guía.
- Obtener la trayectoria que sigue esa guía.
- Generar una serie de instrucciones capaces de dirigir a un vehículo de guiado autónomo (AGV).

2.2. Objetivos funcionales

- El usuario podrá elegir el tipo de binarización, por luminosidad o por color.
- El usuario podrá colocar la cámara en diferente ángulo respecto del suelo, dentro de los margenes de un mínimo de 30 grados y un máximo de 70.
- El usuario podrá usar una plantilla para corregir la distorsión por perspectiva.
- El usuario podrá probar los distintos tipos de binarización fuera del flujo del programa principal, en vista de realizar ajustes y pruebas.

- El usuario podrá ver las distintas fases por las que pasa la imagen, desde que es capturada hasta que se detecta la trayectoria, en vista de realizar ajustes y pruebas.
- El usuario podrá ver las instrucciones de guiado, situación de luminosidad, fotogramas por segundo y situación de trayectoria en el flujo principal del programa.

2.3. Objetivos tecnológicos

- Obtener un flujo de imagen continuo, por lo que necesitaremos un hardware capaz de transmitir de forma continua.
- Obtener un procesamiento de imagen de más de 50 fotogramas por segundo, ya que el AGV requiere de una instrucción cada 2 centésimas de segundo.
- Desarrollo con interfaz en linea de comandos, para hacer la aplicación lo más ligera posible.
- Generar un conjunto de pruebas de todas las funciones creadas y/o probadas.
- Controlar la luminosidad en la imagen para poder reproducir el conjunto de pruebas.
- Utilización de OpenCV para el procesamiento de la imágen.
- Utilización de SciPy para generar funciones mediante regresión lineal.
- Utilización de SciPy para generar funciones mediante regresión lineal.
- Utilización de MatPlotLib para crear ventanas donde ver comparaciones de imágenes.

Conceptos teóricos

3.1. Implementación hardware

En este proyecto trabajaremos con AGVs. Necesitaremos 3 elementos principales para implementarlo:

- Una cámara que capte imágenes, situada adecuadamente en el AGV.
- Una ordenador que ejecute el programa.
- Un elemento para transmitir las instrucciones que genere nuestro programa al sistema de control del vehiculo.

Tras analizar los diferentes elementos y la forma de combinarlos, podemos obtener dos formas de organización de estos elementos, veamos ambas, y sus ventajas e inconvenienetes.

Implementación total en el AGV

En esta implementación se colocarán todos los elementos en el AGV, ejecutando el programa en el ordenador del propio vehículo.

Ventajas

- No se necesita ningún elemento extra.
- La transmisión de la imagen y de las instrucciones es inmediata y por conexión física, lo que asegura una buena transmisión de la información.

Desventajas

- La capacidad de procesamiento del ordenador del vehiculo no es muy grande.
- Esta implementación dificulta el control simultaneo de varios AGVs.

Externalización del ordenador que ejecuta el programa

En esta implementación se externaliza el ordenador donde se ejecuta el programa, a un ordenador fijo, más potente.

Ventajas

- El ordenador externo nos ofrece mayor capacidad de procesamiento.
- Poder ver las instrucciones y situación en tiempo real que se están mandando.

Desventajas

- Se necesita hardware extra, además del que lleva el AGV.
- Necesitamos hardware que nos permita un buen flujo de imágenes hacia el ordenador, y una buena transmisión de las instrucciones hacia el vehículo.

3.2. Procesado de imagen

Para realizar una detección efectiva de la línea guía en la imagen, tenemos que realizar un procesamiento de la imagen, pasandola por distintos filtros para finalmente obtener la trayectoria que representa la línea guía.

Binarización

Binarizar es el proceso que nos permite distinguir ciertas partes o elementos de una imagen aplicando una serie de valores umbrales. La forma en la que se distinguen los elementos es generando una imagen binaria, donde los elementos destacados tienen un valor, y el resto de la imagen tiene otro valor.

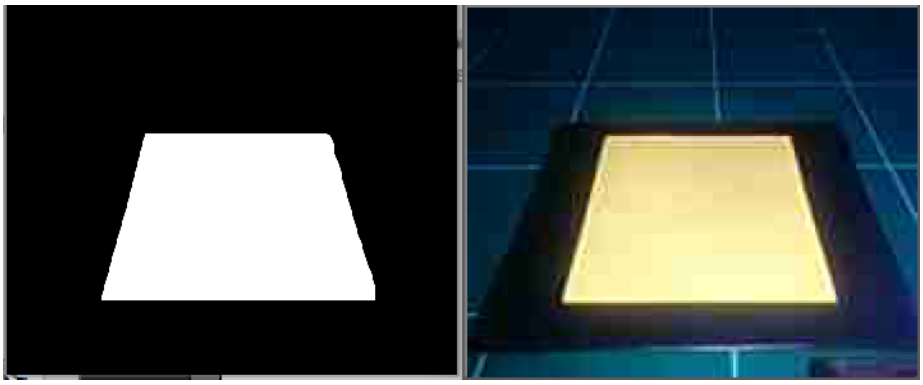

Figura 3.1: Izquierda, imagen binaria. Derecha imagen real.

Primero, tenemos que entender como funciona una pantalla y una cámara, para comprender el formato de imagen más usado actualmente, RGB.

Una imagen RGB está formada por 3 matrices de valores de mismas dimensiones, una matriz para los rojos, otra para los verdes, y otra para los azules (Red-Green-Blue). Este formato, es utilizado actualmente para la representación de imágenes en pantalla, y para la captación de imágenes, ya que se ajusta al formato físico tanto de pantallas como de sensores de cámaras. Veamos algunos conceptos relacionados con esto:

- Pantalla

Una pantalla está formada por diminutos puntos de color, conocidos como pixeles. Cada pixel tiene 3 "luces", una para rojos, otra para verdes y otra para azules. El formato RGB da a cada una de estas luces un valor, que las hará lucir con mayor o menor intensidad.

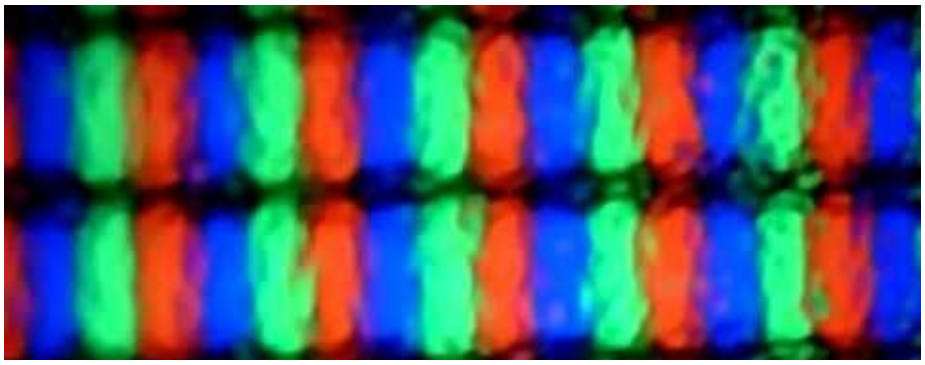

Figura 3.2: Pixels de una pantalla LCD. Fuente: youtube.com

La resolución tanto de la pantalla como de la imagen es un aspecto relevante, ya que influirá en la forma en la que la pantalla representará la imagen.

La resolución de la pantalla es el número de pixels de ancho, por el número de pixels de alto. La resolución de la imagen es el tamaño de la matriz de ancho, por el tamaño de la matriz de alto.

Cuando ambas son iguales, cada valor de la matriz RGB se interpretará en cada uno de los pixeles de la pantalla. En el caso de que sean diferentes, se hará un reescalado de imagen.

Los valores que pueden tomar son desde 0, apagado, hasta 255, intensidad máxima. La combinación de intensidades de las luces RGB formará los distintos colores.

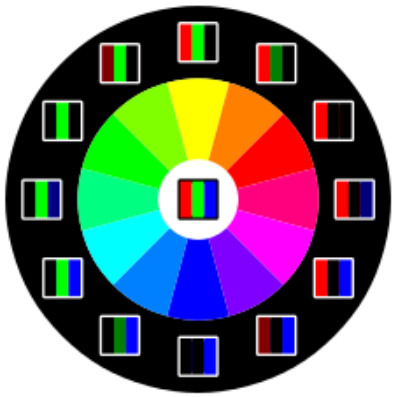

Figura 3.3: Combinación de colores RGB. Fuente: wikipedia.org

Como vemos, tenemos 256^3 combinaciones diferentes, lo que nos da un rango de 16777216 colores posibles.

- Cámara

La forma de funcionamiento de una cámara se basa en un sensor con celdas sensibles a la luz, que permiten registrar la intensidad con la que incide la luz. Una vez más, cada una de estas celdas se corresponderá con un una de las luces de un pixel. Para poder registrar los valores RGB se descompone la luz en sus 3 colores primarios, rojo, verde y azul. Se registra la intensidad de cada color, y se genera la imagen RGB.

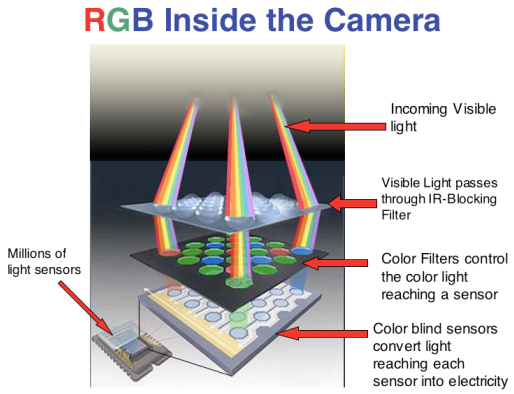

Figura 3.4: Esquema del funcionamiento del sensor de una cámara. Fuente: globalspec.com

Una vez visto el porqué del formato RGB, podemos empezar a explicar los métodos de binarización.

Binarización por luminosidad

Este tipo de binarización consiste en generar una imagen binaria, mediante un umbral aplicado a una imagen donde podamos ver la luminosidad de cada pixel. Para realizar este tipo de binarización necesitamos una imagen en escala de grises, donde solo tenemos luminosidad. Nuestra cámara nos devuelve imágenes RGB, por lo que debemos realizar una conversión.

Esta conversión no es algo trivial, ya que los colores primarios de la luz, RGB, no tienen la misma luminosidad. Por lo que la solución mas simple

que se nos podría ocurrir, realizar la media para cada pixel de sus valores RGB sería errónea.

En su lugar, debemos corregir la luminosidad de cada matriz de color, para igualar estas diferencias de luminosidad que por propia naturaleza los colores primarios tienen.

Para ello, multiplicaremos cada matriz por un valor corrector. Según la documentación de OpenCV[1], la formula que nos permite hacer esta transformación es:

RGB[A] to Gray:$Y = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B$

Donde Y es la imagen resultante.

Una vez obtenemos la imagen en escala de grises, podemos empezar con la binarización.

Para binarizar, primero tenemos que saber cuantos pixels hay de cada valor entre 0 y 255, para tratar de buscar grupos grandes de pixels que compartan valores similares, y así distinguir unos de otros.

Esta información nos la da el **histograma de la imagen**.

El histograma de una imagen es la representación de la distribución de los valores de los pixels de la misma.

En el eje X tendremos de izquierda a derecha valores de 0 a 255. En el eje Y de abajo a arriba valores desde 0 hasta en máximo número de pixels encontrados en ese valor de x.

Figura 3.5: Imagen en escala de grises de la línea guía.

Figura 3.6: Histograma de la Figura 3.5.

Como podemos apreciar, hay dos picos claros en el histograma, uno correspondiente a la línea guía (pixels oscuros, pico izquierdo del histograma), y el resto correspondientes al fondo blanco (pixels claros, pico derecho del histograma), concuerda con la imágen ya que hay mayor cantidad de pixels claros que de oscuros. El umbral tiene que estar entre ambos picos para diferenciar unos pixels de otros.

La forma más simple que se nos puede ocurrir es la de escoger un valor aleatorio entre ambos picos y entonces, recorrer todos los pixels de la imagen en escala de grises, y a cada pixel asignarle un valor si esta por encima del umbral, u otro valor su esta por debajo.

Esta forma, siendo perfectamente valida, es poco recomendable si trabajamos con flujos de datos continuos, como por ejemplo vídeos, o streamings. Normalmente en este tipo de flujos de imágenes, la luminosidad puede variar, por lo que el umbral al ser fijo, puede provocar errores en la binarización.

La solución a esto es usar un umbral dinámico, calculado en función del histograma de cada imagen.

La forma que hemos usado para calcular este umbral ha sido mediante el Algoritmo de Otsu[4], ya implementado en OpenCV.

Lo que hace el algoritmo de Otsu es separar la imagen mediante el umbral en dos zonas. Buscamos que la dispersión dentro de cada segmento sea la mínima (que los pixels dentro de ese segmento se parezcan), pero que entre ambos segmentos sea lo máximo posible.

Para esto, inicialmente calcularemos la media aritmética de los valores de gris de toda la imagen, y después solo de cada zona del histograma. Con estos valores podemos calcular las varianzas de cada zona.

Lo que tenemos que hacer es mantener las variazas de cada zona lo más pequeñas posibles, y conseguir que la varianza entre ambas zonas sea la máxima.

Para conseguir esto, haremos el cociente de la varianza entre las zonas y la suma de las varianzas de cada zona, buscando que este cociente sea el máximo posible.

Veamoslo de una forma más técnica:

$K_0(t)$ y $K_1(t)$ son las zonas del histograma, separadas por el umbral t.

$p(g)$ es la probabilidad del valor gris g, donde g puede ir desde 0 hasta 255, según el formato de escala de grises que estamos usando.

La probabilidad de ocurrencia para cada zona será:

- En la zona que va de 0 a t:
  $P_0(t) = \sum_{g=0}^{t} p(g)$
- En la zona que va de t+1 hasta 255:
  $P_1(t) = \sum_{g=t+1}^{255} p(g)$ o $1 - P_0(t)$

Siendo $\bar{g}$ la media aritmética de los valores de gris para toda la imagen, y $\bar{g_0}$ y $\bar{g_1}$ la media para cada zona, podemos calcular las varianzas de los segmentos:

- En la zona que va de 0 a t:
  $\sigma_0^2 = \sum_{g=0}^{t}(g - \bar{g_0})^2 p(g)$
- En la zona que va de t+1 hasta 255:
  $\sigma_1^2 = \sum_{g=t+1}^{255}(g - \bar{g_1})^2 p(g)$

Ahora necesitaremos las varianzas entre los segmentos, y la suma de la varianza de ambos segmentos.

- Varianza entre segmentos:
  $\sigma_z w^2 = P_0(t) * (\bar{g_0} - \bar{g})^2 + P_1(t) * (\bar{g_1} - \bar{g})^2$
- Suma de las varianzas de cada segmento:
  $\sigma_i n^2 = P_0(t) * \sigma_0^2(t) + P_1(t) * \sigma_1^2(t)$

Por último, nos queda obtener el cocientre de la varianza entre segmentos y la suma de las varianzas de cada segmento.

$$Q(t) = \frac{\sigma_z w^2}{\sigma_i n^2}$$

Este es el cociente que tenemos que maximizar. El umbral será el valor de t.

En OpenCV es sencillo hacer este tipo de binarización, ya que contamos con una función que lo realiza instantáneamente, y nos devuelve tanto el umbral que ha usado, como la imagen ya binarizada.

```
umbral, img_binaria = cv.threshold(img,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
```

Indicamos que queremos hacer la binarización con el algoritmo de Otsu. El valor que se les da a los pixels por debajo del umbral es 0 y el valor por encima es 255. La imagen de origen es img.

El resultado es este:

Figura 3.7: Resultado de la binarización de la Figura 3.5

Como vemos, en esta imagen se puede diferenciar perfectamente la línea guía del resto de elementos de la imagen.

Binarización por color

Inicialmente, en el formato RGB tenemos colores, pero están separados en 3 matrices diferentes, por lo que es complicado poder binarizar de una forma efectiva.

Existe in formato de imagen conocido como HSV[3] (Hue, Saturation, Value), donde la matriz H corresponde con los colores puros de la imagen, la matriz S corresponde a los valores de saturación de cada pixel, y la matriz V corresponde con los valores de negro de cada pixel.

Si representamos esto en un prisma, obtenemos:

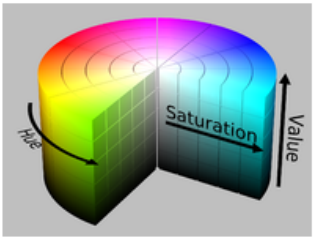

Figura 3.8: Cilindro de colores HSV. Fuente: wikipedia.org

En este tipo de formato tenemos la posibilidad de quedarnos con un pequeño fragmento del cilindro, dividiéndolo según el valor de H, con lo que podríamos binarizar la imagen según un rango de color concreto.

Veamos como pasar de RGB a HSV.

Lo primero que necesitamos es pasar los valores RGB a un entorno en 2 dimensiones. En RGB además de los 3 colores primarios, tenemos las mezclas de colores primarios, amarillo, magenta y cyan. Si a esto añadimos el color blanco puro, y el color negro puro, obtenemos un cubo. Este cubo puede ser representado en 2 dimensiones como un hexágono perfecto, usando como centro del hexágono los vértices de color blanco puro y negro puro. Los valores de RGB serán los que determinen la posición del pixel dentro del cubo, la cual será proyectada en dos dimensiones sobre el hexágono.

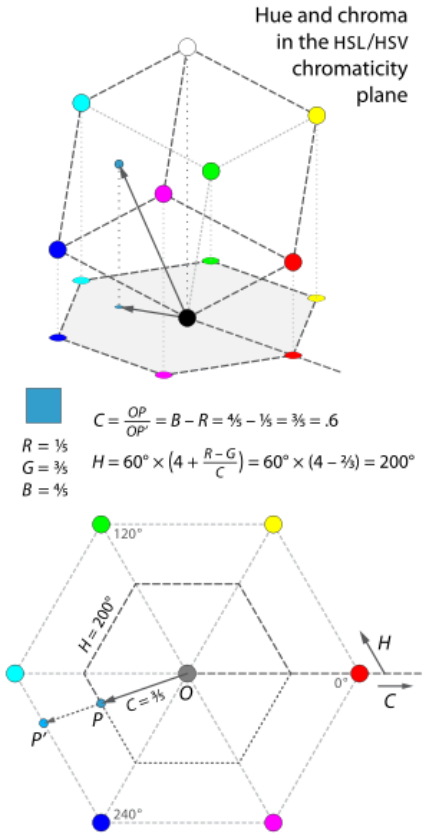

Figura 3.9: Conversión de RGB a HSV. Fuente: wikipedia.org

Con esto podemos obtener el ángulo del color de nuestro pixel, H. Como OpenCV trabaja con matrices de valores enteros de 8 bits, no podemos meter en la matriz valores que salgan del rango $[0, 255]$ por lo que el ángulo será dividido entre 2, pudiendo estar entre 0 y 180.

El valor de negro, V, es el mayor de los valores RGB. Como OpenCV trabaja con enteros de 8 bits, este valor encaja perfectamtente.

El valor de la saturación, S, se calcula a partir del cubo RGB antes mencionado. Suponemos que el hexágono tiene un radio de 0 a 1. Al representar el color RGB en el plano, estará mas o menos alejado del radio del hexagono, este valor se conoce como Chroma, C. Para calcular la saturación, dividiremos C entre V, lo cual nos dará la saturación. Posteriormente convertiremos este valor en escala de 0 a 255, para que OpenCV pueda trabajar con ello.

Repitiendo este proceso para cada uno de los pixels de la imagen, ten-

dremos nuestra imagen en formato HSV.

Una vez tenemos hecho esto, el proceso de binarización es sencillo. Basta con elegir en la rueda de color el ángulo de los colores que queramos. Para el ángulo 0 tenemos el rojo, 60 amarillo, 120 verde, 180 cyan, 240 azul y 300 magenta. Transformaremos el ángulo para que OpenCV lo interprete adecuadamente, simplemente es dividirlo entre 2.

Una vez tengamos el ángulo del color, estableceremos un margen de seguridad[2], para coger todos los pixels con un tono similar. Este margen ha de ser de +10 y -10 para H, y con S y V podemos jugar un poco aunque lo ideal es abarcar el máximo rango posible. Un rango recomendado es entre 100 y 255.

Figura 3.10: Resultado de la binarización por color, escogiendo la gama de amarillos.

En este proyecto se ha simplificado todo este proceso al máximo, ya que es algo complejo. Al usuario se le pedirá calibrar el color mediante unas ventanas de OpenCV, una donde se ve la imagen a color y otra donde se ve el resultado de la binarización. Para elegir un color simplemente tiene que pinchar encima de ese color en la imagen.

Las referencias se incluyen en el texto usando cite [**?**]. Para citar webs, artículos o libros [**?**].

3.3. Transformación de perspectiva

Como hemos visto en secciones anteriores, la principal funcionalidad buscada en este proyecto es el poder ver la línea guía no solo debajo del AGV, también delante, en perspectiva. Para esto simplemente tenemos que

colocar la cámara en una posición determinada, bastante levantada del suelo, y con un ángulo que nos permita ver la máxima perspectiva posible, sin llegar a sobrepasar el horizonte, ya que la información que necesitamos está en el suelo, la línea guía.

El problema de situar la cámara en una posición así, es que no podemos ver las distancias reales en la imagen, puesto que lo estamos viendo todo en perspectiva. Para ver las distancias reales, tendríamos que colocar la cámara paralela al suelo y a una determinada altura, lo cual es complicado. Habría que tener una especie de brazo por delante del robot con la cámara colgando.

Por estos motivos, lo que se usa para resolver este problema es la Transformación de perspectiva. Este tipo de transformación lo que hace es, a partir de una imagen tomada desde una situación en la que la cámara forma un cierto ángulo con el suelo, generar una imagen virtual que se vea totalmente paralela al suelo, podríamos llamarlo una "vista de pájaro".

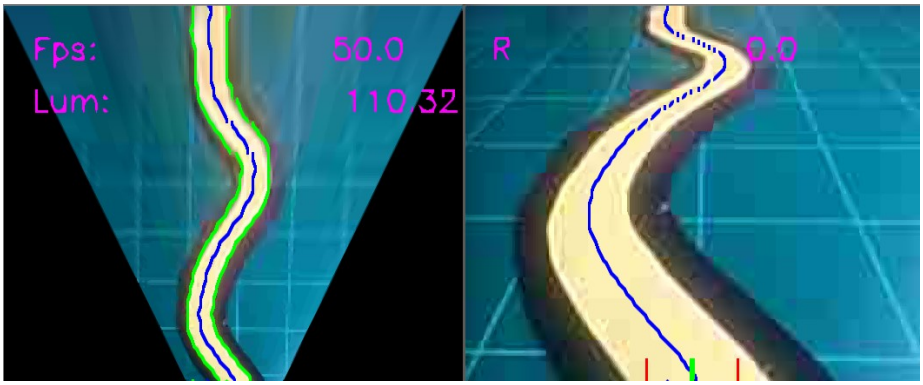

Figura 3.11: Izquierda, vista de pájaro. Derecha, vista original.

3.4. Listas de items

Existen tres posibilidades:

- primer item.
- segundo item.

1. primer item.

Herramientas	App AngularJS	API REST	BD	Memoria
HTML5	X			
CSS3	X			
BOOTSTRAP	X			
JavaScript	X			
AngularJS	X			
Bower	X			
PHP		X		
Karma + Jasmine	X			
Slim framework		X		
Idiorm		X		
Composer		X		
JSON	X	X		
PhpStorm	X	X		
MySQL			X	
PhpMyAdmin			X	
Git + BitBucket	X	X	X	X
MikTEX				X
TEXMaker				X
Astah				X
Balsamiq Mockups	X			
VersionOne	X	X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

2. segundo item.

Primer item más información sobre el primer item.

Segundo item más información sobre el segundo item.

-

3.5. Tablas

Igualmente se pueden usar los comandos específicos de LATEXo bien usar alguno de los comandos de la plantilla.

Técnicas y herramientas

4.1. Técnicas de desarrollo

Metodología ágil

Este proyecto se ha resuelto mediante metodología ágil.

Cabe destacar que este proyecto no es un proyecto software común en el que tenemos unos requisitos iniciales mas o menos claros, y tenemos alguna idea del final. En este caso lo que buscamos es probar diferentes códigos, herramientas, librerías para satisfacer un primer requisito esencial, la detección de lineas en el suelo. Si conseguimos satisfacer ese requisito, podremos proponer nuevos requisitos en base al resultado obtenido.

Entonces, no hay un objetivo final claro, si bien, lo ideal seria avanzar lo máximo posible, también se puede dar el caso de probar y probar diferentes elementos, y no llegar a obtener un resultado esperado.

Por ser un proyecto software, las ventajas de usar una metodología ágil son claras:

- Las personas que participan en el desarrollo adquieren roles diferenciados.
- El desarrollo es incremental, por etapas. Al final de cada etapa se entrega una parte del software funcional al cliente.
- Partimos de unos requisitos básicos y podemos ir añadiendo más a lo largo del desarrollo, o modificar alguno de los existentes.
- La comunicación con el cliente se hace a lo largo de todo el desarrollo, favoreciendo que el producto final sea lo que esperaba.

Una de las metodologías ágiles más usadas es Scrum, cuyo principio clave es que los clientes pueden cambiar de idea sobre lo que quieren y necesitan. En Scrum hay 3 roles diferenciados:

- Product Owner: es el cliente. Propietario de la idea o proyecto a desarrollar.
- ScrumMaster: experto en Scrum, ayuda tanto al Product Owner como al Equipo Scrum a alcanzar los objetivos finales del proyecto.
- Equipo Scrum: equipo de personas que va a realizar el proyecto. Es conveniente que sea un equipo multidisciplinario, en el que cada persona sea experta o conocedora de un campo diferente a las demás. El equipo debe de ser de 5 a 9 personas, para favorecer la comunicación y fluidez del desarrollo.

Se basa en 3 principios básicos, en comparación con metodologías antiguas:

- Favorecer la comunicación del equipo en lugar de la excesiva documentación.
- Buscar la calidad del resultado en los conocimientos de las personas que han participado en el desarrollo, y no en los procesos empleados para hacerlo.
- Solapar las fases de desarrollo en lugar de hacerlas de forma secuencial o en cascada.

Para este proyecto en concreto, el alumno tomara el rol de Equipo Scrum y ScrumManager. El profesor actuará como cliente.

Como antes se ha explicado, este no es un proyecto software común, por lo que partiremos desde un requisito inicial, y cuando el cliente lo consideré valido y completado, propondrá mas requisitos.

En cada sprint se explorarán diferentes herramientas en busca de un objetivo, se harán pruebas con alguna de esas herramientas, estableciendo los parámetros iniciales de cada prueba, el resultado esperado y obtenido. El software evolucionará con cada prueba.

4.2. Herramientas de documentación

LaTeX

Latex es un sistema de composición de texto, que busca la creación de documentos con una alta calidad tipográfica.

Para la documentacion de este proyecto se ha usado TexMaker, que es gratuito bajo la licencia GPL.

Página oficial editor Latex: http://www.xm1math.net/texmaker/

También se ha usado MikTex como implementación de Latex para windows. En su página dicen que por ser un conjunto de paquetes, no tienen una licencia concreta como en otros casos, aun así siguen las directrices de la FSF, y establecen una serie de pautas de uso distribuición y modificación.

Página oficial MikTex: https://miktex.org/

4.3. Herramientas de gestión

Trello

Trello es una herramienta online de gestión de tareas intuitiva y simple. Podemos crear diferentes tableros y crear tareas dentro de estos. La vamos a usar principalmente para comunicarnos entre profesor y alumno. Crearemos 3 tableros, tareas por hacer, tareas en desarrollo y tareas completadas.

Página de trello: https://trello.com/

GitHub

Github es una plataforma online para albergar proyectos desarrollados mediante un sistema de control de versiones git. Es una herramienta mundialmente conocida en el mundo del desarrollo de software, y una de las más usadas para este tipo de propósitos.

Principalmente tiene 3 tipos de elementos:

- Milestones: Son puntos temporales de importancia dentro de la planificación del proyecto.
- Issues: Tareas a realizar. Pueden ser asignadas a um Milestone.

- Commits: Son el elemento que permite ir realizando cambios en el repositorio: añadir elementos, modificar, eliminar... Sirven para ir resolviendo las tareas que vayan surgiendo.

La usaremos para albergar el código del proyecto, y para aplicar metodologías ágiles y scrum. Los Sprints los representaremos con Milestones. Las tareas que vayan surgiendo en cada Sprint como Issues, y la resolución de las diferentes tareas se harán con commits. He elegido esta herramienta por haberla usado en alguna asignatura en la carrera y estar familiarizado con ella, y por ser uno de los repositorios de git más usados.

ZenHub

Es una extension para Google Chrome. Extiende las posibilidades de GitHub, añadiendo elementos como tableros para organizar las tareas, nuevas medidas para evaluar el desarrollo del proyecto y gráficas asociadas a estas medidas. He elegido esta herramienta por ser un gran añadido a GitHub, y por haberla usado en alguna asignatura de la carrera.

4.4. Herramientas de desarrollo

Python

El lenguaje de programación usado para este proyecto es python. Ha sido elegido, primero porque las librerias de OpenCV están para este lenguaje, lo cual es un requisito imprescindible; y segundo por la facilidad de uso, la versatilidad y la gran cantidad de librerias externas con las que cuenta.

Concretamente se ha usado python en su versión 2.7.5. Nos hubiera gustado dar el salto a python 3, pero no hay librerías de OpenCV de forma oficial para esta versión, solo librerías traducidas de python 2 a python 3, por usuarios, por lo que no contamos con un soporte oficial y nos podemos encontrar con errores y sorpresas desagradables.

PyLint

PyLint es una herramienta para analizar y encontrar errores en códigos python. Se puede añadir a IDEs de forma sencilla, y se pueden personalizar los distintos avisos que proporciona. Además permite refactorizar código y cuenta con los estándares para el desarrollo de códigos en python.

OpenCV

OpenCV es una librería software de vision aritificial y machine learning. Nos proporcionará todos los elementos software necesarios para poder obtener y procesar imágenes. OpenCV se distribuye bajo licencia BSD. Esta librería está implementada para varios lenguajes de programación, aunque la elección fue python por la facilidad de uso, la ligereza del lenguaje, y las posibilidades de extension que ofrece.

Visual Studio Code

Visual Studio Coce es el IDE elegido para este proyecto. Es un IDE sencillo, compatible con una grán variedad de lenguajes de programación y compatible con Linux, Windows y MacOS.

Se pueden añadir extensiones de una forma muy sencilla dentro del propio IDE, de hecho es el propio IDE el que nos sugiere instalar algunas extensiones dependiendo del lenguaje de programación que estemos usando.

En mi caso, en cuanto detecto que en mi repositorio había ficheros python, me recomendó instalar la extension de python, la cual incluye pylint, el debugger de python y alguna otra herramienta más.

Además de todo esto, es compatible con git de una forma muy sencilla, simplemente abrimos la carpeta de nuestro repositorio y ya entra en funcionamiento el control de versiones.

Por todo esto, considero que esta herramienta es la más adecuada y la que mejor se adapta a mis requisitos para realizar este proyecto.

Su propietario es Microsoft y sus licencias son:

- Codigo fuente, licencia del MIT.
- Binarios, licencia FreeWare.

IPWebcam

Aplicación android, con dos versiones, gratuita y de pago. Actualmente se usa la versión gratuita. Su funcionalidad es crear un streaming online en directo, accesible desde una url. Es útil puesto que me permite tener la cámara totalmente libre, sin ataduras, pudiéndola mover como desee.

A la hora de realizar pruebas esto es una gran ventaja, en lugar de estar moviendo el ordenador portátil de un lado a otro con la webcam propia del

equipo, o con una cámara conectada por cable; simplemente puedo tener el móvil en un pequeño soporte con ruedas y moverlo libremente a lo largo de la linea. Además me permite tenerlo en diferentes entornos con diferentes luminosidades. Y también hay que tener en cuenta que la cámara de un smartphone es mucho mejor que la de un ordenador portátil, en cuanto a calidad de imagen y captación de luz, por lo que ofrece mayor versatilidad de cara a trabajar en diferentes entornos. Es por esto que he elegido un dispositivo como un smartphone y esta aplicación para la obtención de la imagen con la que voy a trabajar en el proyecto.

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] OpenCV. Color conversions, 2016. [Internet; descargado 23-junio-2018] https://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html.

[2] OpenCV. Changing colorspaces, 2018. [Internet; descargado 23-junio-2018] https://docs.opencv.org/3.4.1/df/d9d/tutorial_py_colorspaces.html.

[3] Wikipedia. Hsl and hsv, 2018. [Internet; descargado 23-junio-2018] https://en.wikipedia.org/wiki/HSL_and_HSV.

[4] Wikipedia. Otsu's method, 2018. [Internet; descargado 23-junio-2018] https://en.wikipedia.org/wiki/Otsu%27s_method.