UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Seguimiento de lineas basado en OpenCV para AGVs Documentación Técnica

Presentado por Antonio de los Mozos Alonso
en Universidad de Burgos — 28 de junio de 2018
Tutor: Jesús Enrique Sierra García

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En esta sección se detallará la planificación del proyecto.

Como planificación se entienden las distintas etapas por las que va a pasar el proyecto hasta su finalización, y la duración de cada etapa. También tenemos que tener en cuenta la viabilidad del proyecto, la posibilidad de que el proyecto sea realizado.

En este proyecto no ha habido una planificación demasiado estricta en cuanto a tiempos de entrega, ya que es un proyecto con una gran componente de investigación. Aún así, se han seguido las directrices de las metodologías ágiles, en concreto dentro del marco de Scrum[6].

La viabilidad económica de una investigación puede llegar a ser difícil de estimar, ya que se intenta agotar los recursos disponibles al máximo, tanto temporalmente como económicamente. Aun así, llegado al punto temporal al que se ha llegado, podemos analizar la viabilidad económica en retrospectiva.

A.2. Planificación temporal

Inicialmente planteamos usar una metodología mas acorde a una investigación, basada en la exploración(etapa de investigación) y evolución(en base a realizar pruebas). Como el producto final ha de ser un software que recoja los resultados de todas las investigaciones y pruebas, finalmente aplicamos una metodología ágil, Scrum. Aun así, dentro de los sprints e iteraciones, es donde se han hecho las etapas de exploración y evolución.

El alumno tomará el papel de equipo de desarrollo y de Scrum Master. El tutor hará de cliente.

Los resultados de las diferentes pruebas realizadas se han ido registrando en el diario de pruebas[1].

Cabe decir que a pesar de que el proyecto estaba planeado para iniciarse en el segundo cuatrimestre, hemos empezado en el primero por motivos académicos del alumno, para poder tener más tiempo de desarrollo.

Sprint 0: 1/10/17-9/11/17

El proyecto parte de un sprint inicial, en el que se explicó el contexto y los elementos con los que se tenía que trabajar. Se preparó del entorno de trabajo, con el hardware y software necesario. Se buscó la mejor forma para realizar el proyecto, explorando dos opciones, trabajar con videos grabados, o trabajar conun streaming en directo.

Tareas

- Preparación de las librerías Python
- Preparación del IDE
- Preparación del Repositorio
- Preparación cámara y aplicación IPWebcam
- Preparación soporte cámara

Backlog

En este sprint lo que hemos hecho es la fase de preparación del proyecto, preparar todos los materiales. También se dieron los conocimientos básicos al alumno en reuniones con el tutor, sobre el contexto de trabajo, que se pretendía realizar, que se pretendía probar y que se pretendía conseguir.

Investicación

En esta fase se investigaron las formas de organizar los elemtnos con los que se iba a trabajar. Inicialmente se propuso usar la WebCam integrada en el portátil, pero finalmente se optó por usar un smartphone que retransmitiera vídeo, ya que nos da mucha mayor libertad.

Esto podría asociarse a las formas de organización de los diferentes elementos (cámara, ordenador y sistema de control del AGV) vistas en la memoria. Resaltar que para este proyecto solo hemos trabajado con la cámara y el ordenador, puesto que esta enfocado a la detección de líneas, no al control de un AGV.

Pruebas

Se realizaron las pruebas de conexión pertinentes. Inicialmente se usaba el router de la red local de casa, con el smartphone y el ordenador conectados vía wifi. Al ser la conexión inalámbrica, y el router estar recibiendo peticiones de otros dispositivos, había más latencia y perdida de paquetes, lo que generaba un mal rendimiento de la transmisión de vídeo.

A si que finalmente, se optó por usar un router especifico para este sistema, conectando por cable el ordenador para optimizar al máximo la conexión y evitar la perdida de paquetes, y conectado vía wifi al smartphone.

Sprint 1: 9/11/17-23/11/17

En el primer sprint realizado se buscó la forma de obtener la tasa de Fps necesaria para cumplir con los requisitos establecidos, así como realizar una prueba de binarización de la línea guía, hecha de unos materiales concretos, buscar diferentes algoritmos, probarlos, generar una funcionalidad para medir la luminosidad y así establecer los parametros bajo los que se realizan las pruebas.

Tareas

- Obtener una tasa de Fps que permitiera realizar el procesado de imágenes en menos de 2 centesimas de segundo.
- Crear la plantilla de fondo blanco y línea negra para hacer la binarización por luminosidad.
- Buscar diferentes algoritmos de binarización por luminosidad.
- Probar los diferentes algoritmos, dejar registro de las pruebas, y elegir el que mejor funcione.
- Generar una funcionalidad para medir la luminosidad en la imagen.
- Crear un archivo donde ir implementando todas las funciones probadas, con los parámetros necesarios.

Backlog

En este sprint hemos realizado los ajustes pertinentes en la cámara para cumplir con las restricciónes de Fps, hemos buscado, probado y registrado diferentes algoritmos de binarización, y finalmente hemos dejado una implementación con el mejor algoritmo encontrado. Además para realizar las pruebas se ha necesitado una funcionalidad que permita medir la luminosidad de la imagen, por lo que también se ha implementado esta funcionalidad.

Investicación

Se han investigado los diferentes algoritmos de binarización por luminosidad.

Pruebas

Se han realizado pruebas con todos los algoritmos, estableciendo parámetros y dejando registros de los resultados.

Sprint 2: 23/11/17-14/12/17

En el segundo sprint buscamos el tomar medidas de la línea sobre la imagen, para comprobar la eficacia de los metodos de binarizacion, se probaron diferentes funciones que median el ancho real y el ancho calculado de la línea, estando la línea en perpendicular a la cámara.

Tareas

- Diseñar función que permita realizar medidas de la línea sobre la imagen.
- Diseñar función para estimar el ancho de la línea en perspectiva.
- Probar las funciones, y comprobar el fallo en la medición de la anchura en diferentes condiciones de luminosidad.

Backlog

Se implementaron las funciones necesarias para medir la línea normal y en perspectiva, estando perpendicular a la cámara. Aunque estas funciones son correctas, la funcionalidad para resolver la distorsión de perspectiva es muy limitada, funcionando solo cuando la línea esta en perpendicular a la cámara.

Investicación

Primeros inicios en la resolución de la distorsión de perspectiva, buscando el saber el ancho real de la línea en diferentes zonas de la imagen.

Pruebas

Se realizaron pruebas de medición de la línea y comparación con la anchura real calculada.

Sprint 3: 12/2/18-30/04/18

En el tercer sprint buscamos tomar medidas de una forma más avanzada, no solo si la línea esta de forma perpendicular a la cámara, sino estando en cualquier posición. Para esto necesitaremos entrar en la problemática de la perspectiva. Además iniciaremos la investigación del sistema de guiado a través de la trayectoria.

Tareas

- Avanzar en la resolución de distorsión por perspectiva
- Implementar una funcionalidad que permita obtener la vista de pájaro.
- Comprobar que la imagen en vista de pajaro guarda las mismas proporciones que la realidad.
- Implementar una funcionalidad que permita obtener la trayectoria (centro de la línea).
- Indagar en los sistemas de guiado, realizando una funcionalidad que calcule la distancia de la trayectoria al centro de la imagen.

Backlog

En este sprint se hacen avances muy importantes, sobre todo la resolución total de la distorsión por perspectiva, problemática principal del proyecto. Se consigue generar una funcionalidad que genera la vista de pájaro guardando las proporciones reales. Se obtiene la trayectoria mediante los bordes de la línea. Iniciamos el desarrollo del sistema de guiado.

Investicación

Se investigan formas de resolver la distorsión por perspectiva adecuadamente, así como de relacionar la funcionalidad que ya tenemos con esta problemática. Se investigan los diferentes tipos de sistema de guiado.

Pruebas

Las pruebas realizadas en el ámbito de la distorsión por perspectiva consisten principalmente en enganchar las diferentes funcionalidades que tenemos y que funcionen en conjunto, y comprobar que la vista de pajaro guarda las proporciones de la realidad, para esto medimos ángulos. Dentro de los sistemas de guiado, hacemos una prueba que nos permite ver como actúa el sistema de guiado implementado.

Sprint 4: 30/04/18-28/06/18

En el cuarto sprint buscamos nuevas formas de binarizacion, en concreto la binarizacion por color. Crear una interfaz para dar una funcionalidad central al proyecto, y finalizar el sistema de guiado y su forma de representación. Además se planteó la posibilidad de integrar las librerías de canKin para conectar el sistema de guiado básico con un robot, si la evolución temporal lo permitía.

Tareas

- Indagar en la binarización por color.
- Implementar un sistema de binarización por color compatible con el resto de funcionalidad.
- Implementar una interfaz para dar una cohesión a todas las funcionalidades implementadas.
- Mostrar en la interfaz toda la información de los algoritmos.
- Integrar librerías canKin, opcional, en función de la evolución del resto de tareas.
- Generar ejecutable de la aplicación para hacer el release[7].

Backlog

Se cumplieron todas las tareas obligatorias, tanto implementar un sistema de binarización por color como la interfaz que conecta todas las funcionalidades implementadas. Además se consiguió mostrar la información en las propias ventanas de OpenCV sin sacrificar rendimiento de la aplicación. Se hizo el release de la aplicación en el repositorio.

Investicación

Se investigó acerca de los algoritmos de binarización por color, dando lugar a la implementación de un sistema de binarización por color.

Pruebas

Las pruebas realizadas parten desde los distintos materiales que se podían usar para la binarización por color. Inicialmente se uso cartulina amarilla, no funcionaba adecuadamente hasta que remarcamos su borde con cinta aislante negra, lo cual resaltó el borde e hizo que funcionara bien.

A.3. Estudio de viabilidad

Necesitamos que el proyecto sea tanto viable económicamente, es decir, que sea rentable; y que sea viable legalmente, es decir, que no viole alguna ley a lo largo del transcurso del mismo.

Viabilidad económica

En este apartado se analizarán los costes del proyecto.

Costes de personal

Consideraremos que este proyecto se ha realizado por una persona a lo largo de seis meses: Octubre, Noviembre, Febrero, Marzo, Abril y Mayo. Trabajando 20 días al mes, 4 horas diarias. El sueldo de un programador sin experiencia actualmente son 10 euros por hora. Haciendo cálculos: $10 euros/hora * 4 horas/dia = 40 euros/dia$ $40 euros/dia * 20 dias/mes = 800 euros/mes$ $800 euros/mes * 6 meses = 4800 eurosdesalario$

Costes de hardware

En esta sección calcularemos los costes hardware del proyecto. Incluiremos todos los elementos hardware empleados. Consideraremos la vida útil del hardware en 6 años. La amortización la calcularemos como:

$Amortización = (Valordelbien)/Vidautil$

- Ordenador portátil: 500 euros.
  $500euros/(12 * 6)meses = 6{,}94euros/mes$
  Por tanto en 6 meses:
  $6,94euros/mes * 6meses = 41{,}64eurosbrutos$
- Smartphone: 300 euros. $300euros/(12 * 6)meses = 4{,}16euros/mes$
  Por tanto en 6 meses:
  $4{,}16euros/mes * 6meses = 24{,}96eurosbrutos$
- Router: 50 euros. $50euros/(12 * 6)meses = 0{,}69euros/mes$
  Por tanto en 6 meses:
  $0{,}69euros/mes * 6meses = 4{,}14eurosbrutos$

Costes de software

La única licencia no gratuita usada ha sido el sistema operativo Windows 10 Home, que tiene un coste de 120 euros. Estimamos su vida útil en 3 años.

$120euros/(12 * 3)meses = 4{,}32euros/mes$

Por tanto en 6 meses:

$4{,}32euros/mes * 6meses = 25{,}92eurosbrutos$

Costes de materiales

Aquí sumaremos los costes de la realización de plantillas. Estos son costes sin amortización, puesto que las plantillas una vez creadas no sirven para nada más.

- Coste de las cartulinas: 10 euros.
- Coste de la cinta aislante: 5 euros.

Además hemos creado un soporte con Lego, con un coste aproximado de 5 euros.

Costes de totales

Vamos a estimar los costes del trabajador según los Tipos de Cotización Régimen General Ejercicio 2018[5].

Porcentajes de cotización:

- Contingencias comunes: 23.6.
- Desempleo: 5.5
- 0.6

Realizando la suma, tenemos un porcentaje de: 29.7

Por tanto el coste real del trabajador serán: $4800 + (4800 * 29{,}7) = 4942{,}56 euros$

Sumando el resto de costes: $4942{,}46 + 41{,}64 + 24{,}96 + 4{,}14 + 25{,}92 = 5039{,}12 euros$

Beneficio

En este apartado estimaremos los beneficios que se podrían obtener de este software. EL software por si solo no se puede distribuir, se ha de distribuir junto con el vehículo que tiene que guiar.

Suponemos que esta herramienta de guiado se implementa en 300 vehículos y que el coste añadido por cada vehículo que lo implemente es de 20 euros anuales.

$300 vehiculos * 20 euros/vehículo = 6000 euros/año$

Recuperaríamos la inversión en:

$5039{,}12/6000 euros anuales = 0{,}83 años$

A partir de ese tiempo, unos 10 meses, obtendríamos beneficios.

Viabilidad legal

En este apartado veremos las licencias del software, compatibilidades entre ellas y la licencia final de la aplicación. También, aunque no se ha trabajado con ello, veremos la legislación de robots y vehículos autónomos en España, ya que el software sí esta pensado para dirigir a un vehículo de forma autónoma.

Software

Primero veamos las licencias del software usado:

- OpenCV, licencia BSD
- SciPy, licencia BSD
- Numpy, licencia BSD
- MatPlotLib, licencia BSD
- Urllib, licencia MIT

Según la guía de licencias para python[3], la licencia adecuada para este proyecto podría ser MIT o BSD, ya que son compatibles entre ellas, y cumplen con todas las licencias de las librerías usadas.

En este caso elegiremos licencia MIT ya que es menos restrictiva en cuanto a los avisos de Copyright, y en cuanto a publicitar del software. Con la licencia BSD se tiene que remarcar siempre que se use el software usado, quien lo creó en toda la publicidad que se haga.

La licencia Mit permite[2]:

- Uso de la herramienta
- Distribución de la herramienta
- Distracción y modificación
- Aviso de copyrigth y garantía solo en el software que lleve partes del software marcado con esta licencia. En publicidad y otros elementos no haría falta.
- Mezclar la herramienta con otro software.
- Vender la herramienta
- Asignarle una licencia más restrictiva.
- Publicarla

La aplicación IpWebcam no la consideramos puesto que aunque ha sido usada para el desarrollo, en la versión final del producto, lo único que tenemos que suministrar al programa es una url del servidor de vídeo, por lo que el usuario podrá usar la aplicación o sistema que necesite.

Legislación de robots autónomos

Actualmente en España no hay una legislación clara sobre robots y vehículos autónomos, aun así dentro del marco Europeo ya se está trabajando en algunas normas o objetivos que en un futuro no muy lejano pueden afectar a este software[4]

- Necesidad de un registro de robots.
- Necesidad de un sistema de responsabilidad objetiva por daños causados y un sistema de seguro obligatorio.
- Necesidad de fijar un régimen de cotización a la seguridad social, por los dueños de los robots.

Al final estas normas son previsibles, el trabajo que está haciendo un AGV de forma autónoma lo podría hacer un trabajador que cotiza y que puede cometer fallos y generar daños.

El registro de robots se tendrá para no tener robots "trabajando en negro", al igual que las personas.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apartado describiremos los requisitos que ha de cumplir nuestra herramienta en cuanto a funcionalidad.

B.2. Objetivos generales

Los objetivos generales de la herramienta son:

- Detección de lineas
- Trabajar en perspectiva
- Implementar un sistema de guiado

B.3. Catalogo de requisitos

Requisitos Funcionales

- R1. Probar sistema de binarización por luminosidad.
- R2. Probar sistema de binarización por color.
- R3. Probar el sistema de resolución de distorsión de perspectiva.
- R4. Calibrar binarización por color.

- R5. Calibrar distorsión de perspectiva.
- R6. Calibrar zona segura del sistema de guiado.
- R7. Calibrar angulo de giro del sistema de guiado.
- R8. Ejecutar el flujo principal del programa.
- R9.Solicitar URL del servidor de vídeo.

Requisitos no Funcionales

- El software ha de conseguir la máxima tasa de Fps, idealmente 50.
- El software necesita de un servidor de vídeo que le proporcione imágenes mediante una url.
- El software puede ser extendido, por lo que el código debe de ser modular.

B.4. Especificación de requisitos

R1. Probar sistema de binarización por luminosidad.

Versión V1.0

Autor Antonio de los Mozos Alonso

Descripción Probar el sistema de binarización por luminisidad fuera del flujo principal para ver que materiales o forma de iluminación es la adecuada en el entorno.

Precondición Haber obtenido una conexión de vídeo, Requisito R9.

Secuencia Normal Paso y Descripción

1. El usuario selecciónа la opción en el menú principal.
2. El programa abre una ventana donde se ve la imagen real y la imagen binarizada.
3. El usuario comprueba que la conexión de vídeo es adecuada, se ve sin cortes.
4. El usuario prueba sus materiales y condiciones de luminosidad.

5. Pulsar la tecla S para salir.

Postcondición El usuario consigue ver que materiales y luminosidad funcionan mejor para este tipo de binarización.

Excepciones Luminosidad no sea lo suficientemente grande como para distinguir algo en la imagen.

Importancia Media.

Comentarios Ninguno.

R2. Probar sistema de binarización por color.

Versión V1.0

Autor Antonio de los Mozos Alonso

Descripción Probar el sistema de binarización por color fuera del flujo principal para ver que materiales o forma de iluminación es la adecuada en el entorno.

Precondición Haber obtenido una conexión de vídeo, Requisito R9.

Secuencia Normal Paso y Descripción

1. El usuario selecciona la opción en el menú principal.
2. El programa abre una ventana donde se ve la imagen real y la imagen binarizada.
3. El usuario comprueba que la conexión de vídeo es adecuada, se ve sin cortes.
4. El usuario prueba sus materiales y condiciones de luminosidad.
5. Pulsar la tecla S para salir.

Postcondición El usuario consigue ver que materiales y luminosidad funcionan mejor para este tipo de binarización.

Excepciones Luminosidad no sea lo suficientemente grande como para distinguir algo en la imagen.

Importancia Media

Comentarios Ninguno

R3. Probar el sistema de resolución de distorsión de perspectiva.

Versión V1.0

Autor Antonio de los Mozos Alonso

Descripción El usuario necesita ver las fases por las que pasa la imagen hasta que se resuelve la distorsión de perspectiva.

Precondición Haber calibrado el sistema de binarización Requisito R4 o Requisito R10. Haber calibrado la distorsión de perspectiva,Requisito R5.

Secuencia Normal Paso y Descripción

1. El usuario selecciona la opción en el menú.
2. El programa solicita el tipo de binarización y todas las calibraciones necesarias.
3. EL programa abre una ventana donde se ven las fases por las que pasa la imagen.
4. El usuario realiza pruebas en el entorno para ver que materiales, ángulo de la cámara y luminosidad le conviene.
5. El usuario sale de la ventana haciendo click en el aspa roja.

Postcondición Permite que el usuario vea las fases por las que pasa la imagen, y permite realizar ajustes en el entorno según le convenga.

Excepciones No haber calibrado bien los sistemas necesarios.

Importancia Media.

Comentarios Ninguno.

R4. Calibrar binarización por color.

Versión V1.0

Autor Antonio de los Mozos Alonso

Descripción Calibrar el sistema de binarización por color para usarlo en el flujo del programa principal.

Precondición Haber realizado las pruebas pertinentes para comprobar que el algoritmo va a funcionar bien en el entorno. R2.

Secuencia Normal Paso y Descripción

1. Cuando se requiera realizar la calibración se abrirá una ventana de OpenCV, con la imagen en color y la imagen binarizada.
2. Pinchar sobre el color por el que queramos binarizar.
3. Pulsar la tecla S para salir.

Postcondición El sistema de binarización por color quedará calibrado para ser usado en el flujo principal.

Excepciones Luminosidad no sea lo suficientemente grande como para distinguir algo en la imagen.

Importancia Alta.

Comentarios Ninguno.

R5. Calibrar distorsión de perspectiva.

Versión V1.0

Autor Antonio de los Mozos Alonso

Descripción Permite calibrar el sistema de resolución de distorsión por perspectiva.

Precondición Necesitamos tener una plantilla cuadrada, y que el sistema de binarización la detecte adecuadamente.

Secuencia Normal Paso y Descripción

1. El sistema pedirá que se calibre la distorsión de perspectiva.
2. El sistema solicitará que se coloque la plantilla del cuadrado delante de la imagen.
3. El sistema abrirá una ventana OpenCV donde se verá la imagen binarizada.
4. El usuario tendrá que colocar el cuadrado en la parte baja de la imagen, lo más centrado y paralelo a la cámara posible.
5. El usuario pulsara S para salir de la ventana.

6. El sistema pedirá al usuario que retire la plantilla del cuadrado.
7. El sistema mostrará en la linea de comandos el coeficiente calculado asociado al ángulo.
8. Se calibrará el sistema de resolución de distorsión de perspectiva.

Postcondición El sistema de resolución de distorsión de perspectiva quedará calibrado.

Excepciones Colocar el cuadrado de una forma no adecuada, dando lugar a calibraciones erróneas.

Importancia Alta.

Comentarios Ninguno.

R6. Calibrar zona segura del sistema de guiado.

Versión V1.0

Autor Antonio de los Mozos Alonso

Descripción El sistema de guiado cuenta con una zona segura que se tiene que calibrar.

Precondición Haber obtenido un flujo de video. Requisito R9.

Secuencia Normal Paso y Descripción

1. El sistema solicitará un valor numérico decimal por linea de comandos, entre 0 y 0.3.
2. El usuario escribirá el valor que desee y pulsará enter.
3. El sistema validará el valor y continuará.

Postcondición La zona segura del sistema de guiado se generará.

Excepciones

Importancia Alta

Comentarios Ninguno.

R7. Calibrar angulo de giro del sistema de guiado.

Versión V1.0

Autor Antonio de los Mozos Alonso

Descripción Permite establecer el ángulo de giro máximo que puede realizar el vehículo, para calibrar el sistema de guiado para ese vehículo.

Precondición Haber establecido la zona segura del sistema de guiado. Requisito R6.

Secuencia Normal Paso y Descripción

1. El sistema solicitará un valor numérico decimal por linea de comandos, entre 0 y 90.
2. El usuario escribirá el valor que desee y pulsará enter.
3. El sistema validará el valor y continuará.

Postcondición El sistema de guiado quedará calibrado para ese tipo de vehículo.

Excepciones Si no se calibra bien, el vehículo puede tener problemas a la hora de realizar giros.

Importancia Alta

Comentarios Ninguno.

R8. Ejecutar el flujo principal del programa.

Versión V1.0

Autor Antonio de los Mozos Alonso

Descripción Poder ejecutar el flujo del programa principal con todos los sistemas de guiado y resolución de distorsión por perspectiva.

Precondición Haber calibrado bien el sistema de binarizacion, el sistema de resolución de distorsión por perspectiva y el sistema de guiado. Requisito R7, Requisito R6, Requisito R5, Requisito R4 y Requisito R9.

Secuencia Normal Paso y Descripción

1. El usuario elegirá la opción en el menú principal.
2. El sistema pedirá hacer la calibración de binarización por color, si se usa.
3. El sistema pedirá hacer la distorsión de perspectiva.
4. El sistema pedirá hacer la calibración de la zona segura del sistema de guiado.
5. El sistema pedirá hacer la calibración del angulo de giro del sistema de guiado.
6. El sistema abrirá una ventana de OpenCV donde se mostrará el sistema de guiado y la resolución de distorsión por perspectiva.

Postcondición El sistema comenzará a generar instrucciones de guía.

Excepciones Alguna de las calibraciones no se haya hecho de forma adecuada.

Importancia Alta.

Comentarios Ninguno.

R9.Solicitar URL del servidor de vídeo.

Versión V1.0

Autor Antonio de los Mozos Alonso

Descripción Al inicio del programa se nos solicitará la URL de donde vamos a obtener el vídeo.

Precondición Haber iniciado el programa.

Secuencia Normal Paso y Descripción

1. El programa solicitará una URL de video.
2. El usuario introducirá por linea de comandos la URL.
3. El programa validará que tiene conexion a la URL.

Postcondición Se establecerá una conexión con el servidor de video.

Excepciones No poderse conectar con el servidor de video.

Importancia Alta.

Comentarios Ninguno.

B.5. Casos de uso

Diagrama de casos de uso de la herramienta:

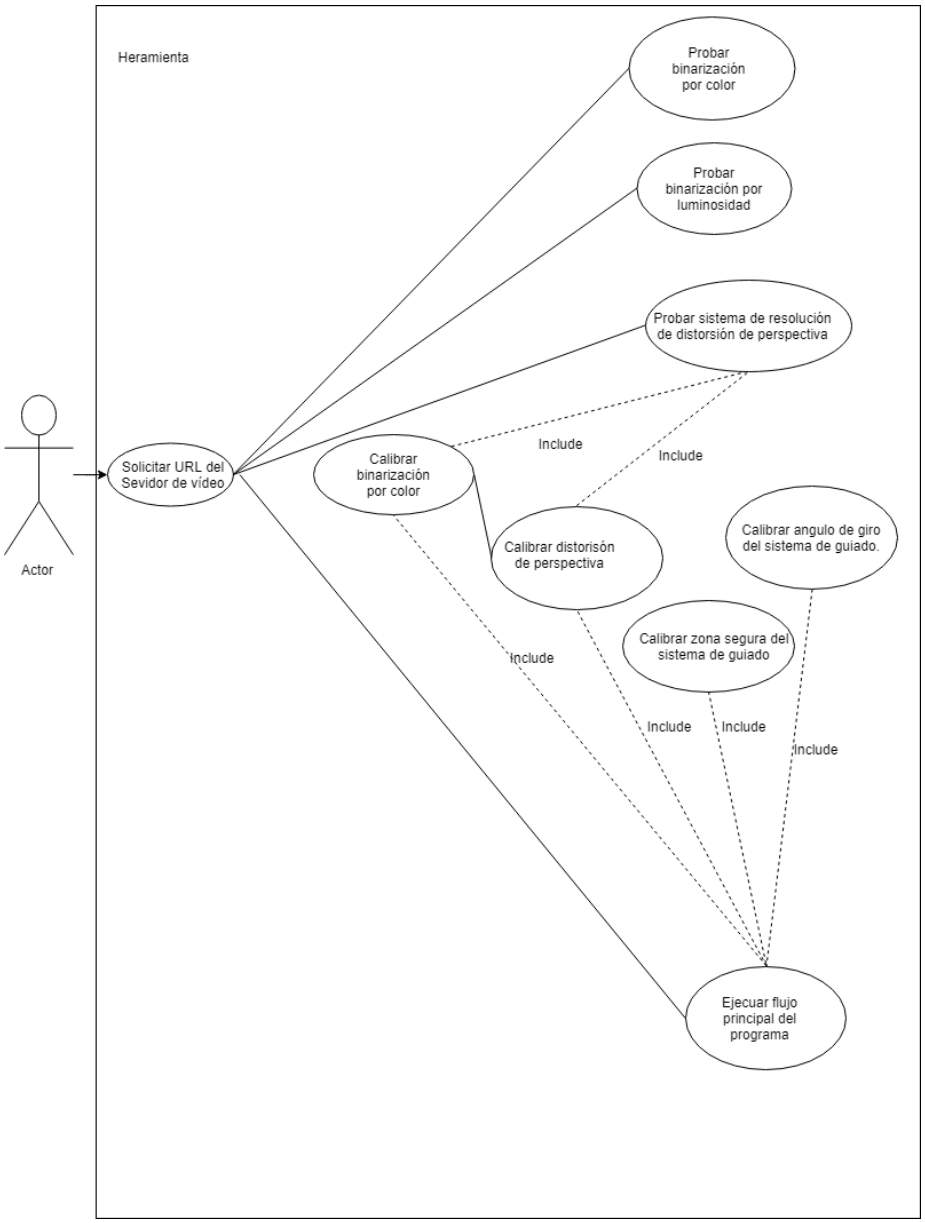

Figura B.1: Diagrama de casos de uso.

Apéndice C

Especificación de diseño

C.1. Introducción

En este apartado veremos como se ha diseñado la herramienta.

C.2. Diseño de datos

El único dato relevante con el que trabajamos son imágenes. Veamos los formatos de imagen usados:

- Las imágenes se generan en formato RGB, el cual tiene 3 matrices de enteros de 8 bits, una para Rojos, Una para Verdes y otra para Azules.
- Las imágenes en formato BGR, tienen 3 matrices de enteros de 8 bits, una para Azules, una para Verdes y otra para Rojos.
- Las imágenes en escala de grises tiene una sola matriz de valores enteros de 8 bits.
- Las imágenes en formato HSV tienen 3 matrices de enteros de 8 bits, una para los colores, H; otra para la saturación, S; y otra para el valor de negro, V.

Las transformaciones entre formatos se harán con llamadas a OpenCV.

C.3. Diseño procedimental

A continuación veremos la interacción entre los elementos del programa.

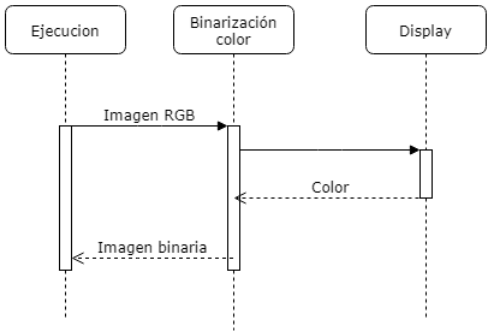

Figura C.1: Interacción. Prueba de binarización por color.

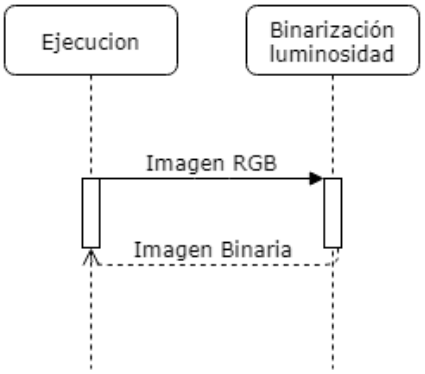

Figura C.2: Interacción. Prueba de binarización por luminosidad.

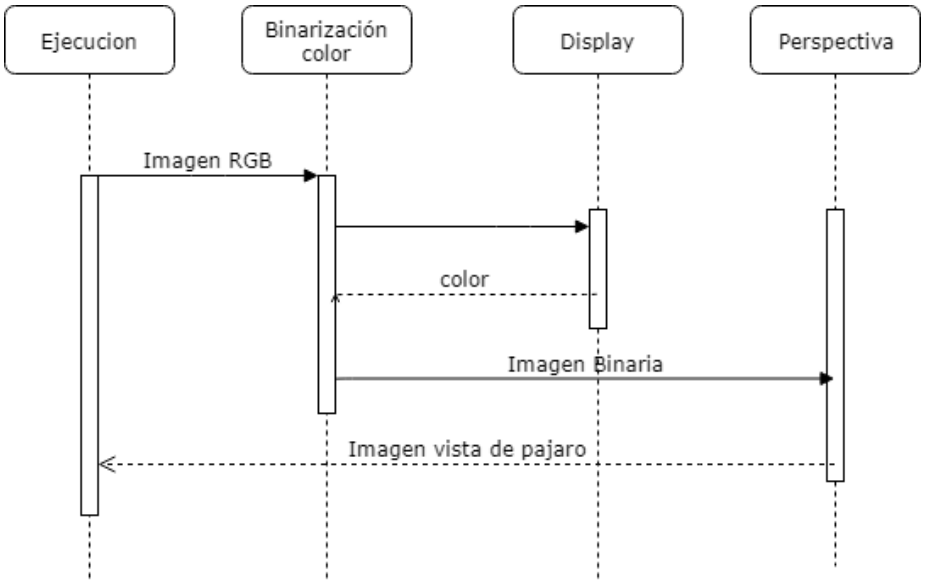

Figura C.3: Interacción. Probar perspectiva con binarización por color.

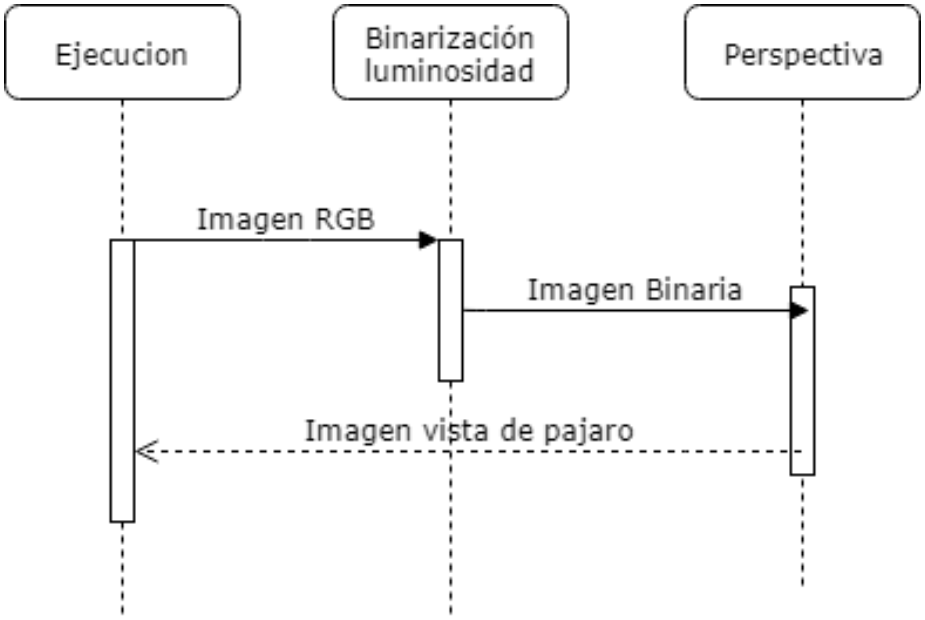

Figura C.4: Interacción. Probar perspectiva con binarización por luminosidad.

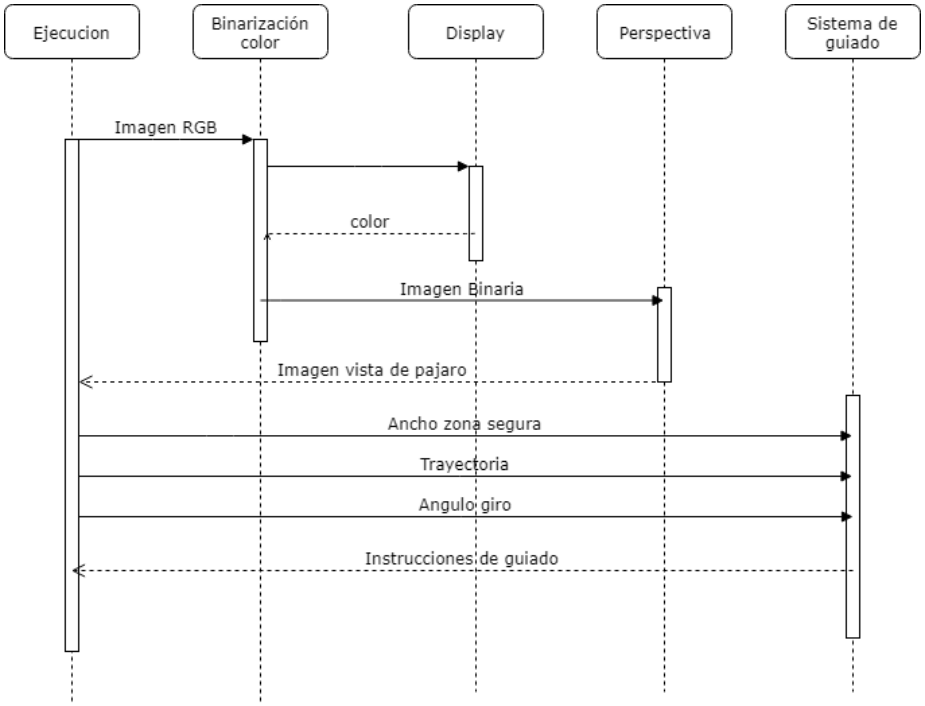

Figura C.5: Interacción. Ejecución principal con binarización por color.

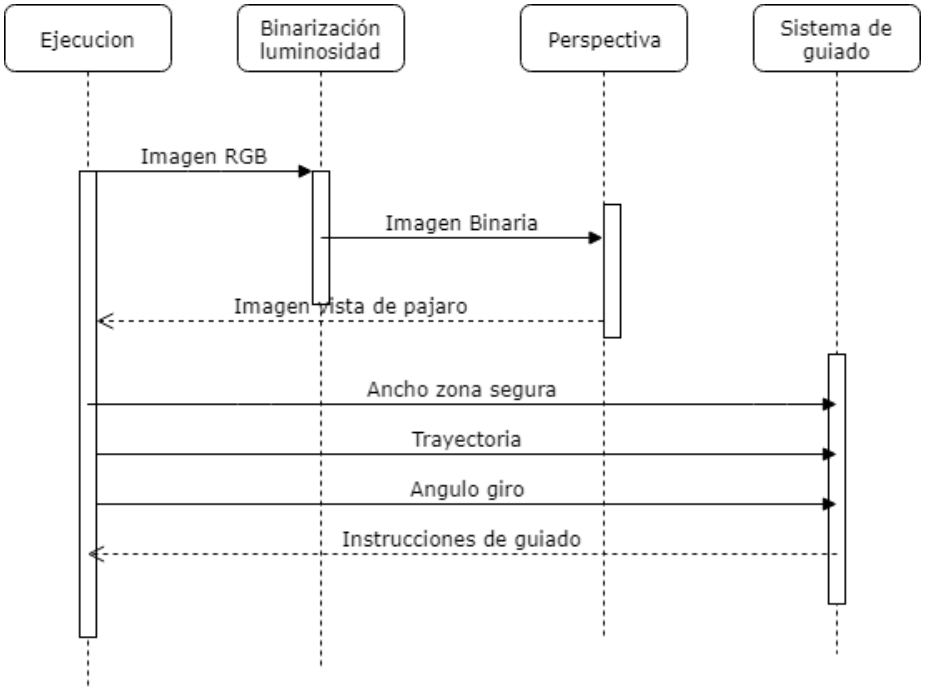

Figura C.6: Interacción. Ejecución principal con binarización por luminosidad.

C.4. Diseño arquitectónico

A continuación se mostrará la organización de los diferentes módulos del programa.

Destacar que en el diseño del programa se han usado clases y scripts, remarcados con la palabra (Script) al lado de su nombre en el diagrama.

El script ejecucion es donde está el programa principal

El script toolbox es donde estan la mayoría de funciones de procesamiento de imágenes. Ha sido pensado para poder ser reutilizado en otros proyectos.

El resto de clases, tienen dependencias con la clase webcam_stream, la clase que proporciona el streaming de vídeo.

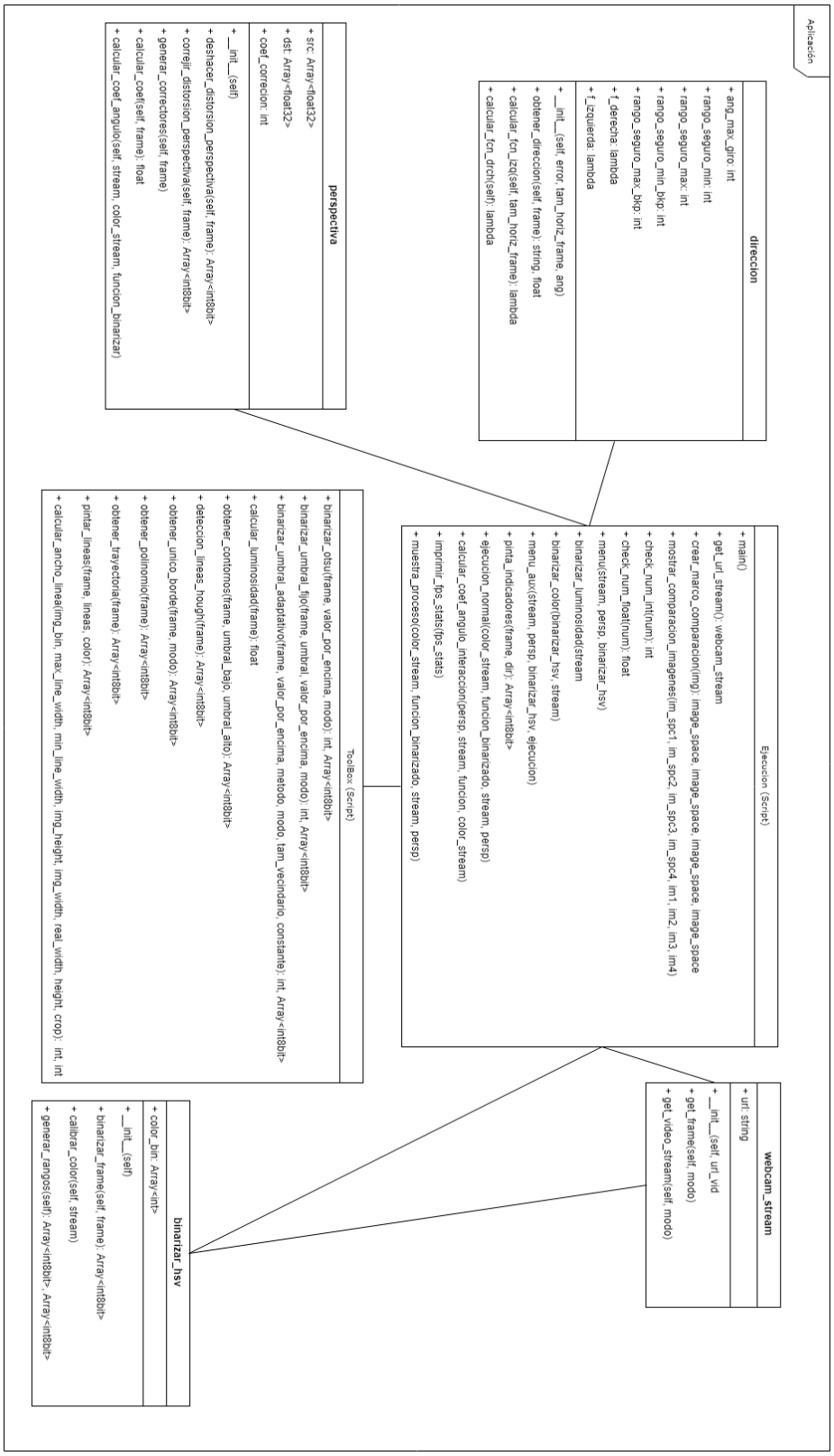

Figura C.7: Diagrama de clases y scripts.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apartado mostraremos todas las herramientas necesarias para ponernos a trabajar con este proyecto.

El proyecto es accesible desde: https://github.com/ama0114/TFG-OpenCV/

D.2. Estructura de directorios

Los scripts y clases están dentro de la carpeta src.

Listado de archivos:

- binarizar_hsv.py
- direccion.py
- ejecucion.py
- ejecucion.spec
- toolbox.py
- webcam_stream.py
- perspectiva.py

El archivo ejecucion.spec se usa con la herramienta pyinstaller para generar el ejecutable.

En la carpeta pruebasPython encontraremos archivos de prueba.

Listado de archivos:

- holamundo.py
- pruebaVideo.py
- test_angulo.py
- tes_luminosidad.py
- recIzq.avi

Los archivos test_angulo.py y test_luminosidad.py requieren algunos de los archivos de la carpeta src, por lo que para hacerlos funcionar los tenemos que mover ahí. Se han colocado en la carpeta de pruebas para estructurar bien el repositorio.

D.3. Manual del programador

Ahora veremos como preparar el entorno de trabajo.

Python

Lo primero que haremos será instalar Python. La versión utilizada para este proyecto es la 2.7.15. Lo podemos descargar en: https://www.python.org/downloads/release/python-2715/

Una vez hecho esto instalaremos pip para Python. Necesitaremos el archivo get_pip.py, lo podemos descargar en: https://bootstrap.pypa.io/get-pip.py

Para instalar pip, abrimos un terminal, nos colocamos mediante el comando cd en el directorio donde tengamos el archivo getpip.py y ejecutamos: python getpip.py

Una vez que tengamos pip, instalaremos las librerías necesarias, para ello usaremos los comandos:

- pip install opencv-python

- pip install matplotlib
- pip install numpy
- pip install scipy
- pip install pylint
- pip install pyinstaller

Con las primeras 4 liberias podremos ejecutar el código desde python, pylint es un parser para python, que nos ayudará en la labor de programación y pyinstaller nos servirá para crear el ejecutable.

Se ha dejado en el repositorio un archivo requirements.txt para instalar todas las librerías con un solo comando. Nos descargamos el archivo, abrimos una consola de comandos, nos desplazamos con el comando cd hasta donde tengamos el archivo y ejecutamos el comando: pip install -r requirements.txt

IDE

Una vez preparado Python, ya nos podríamos poner a programar desde el propio IDE de Python.

Aun así, en este proyecto hemos usado Microsoft Visual Studio Code.

Lo podemos descargar desde aquí: https://code.visualstudio.com/

La instalación es como la de cualquier programa para Windows.

Inicialmente este IDE ya viene preparado para trabajar, aun así, si necesitamos realizar alguna configuración, la podemos hacer desde: Archivo, Preferencias, Configuración.

Se nos abrirá una nueva ventana del editor con el archivo de configuración, aquí podemos realizar la configuración que necesitemos.

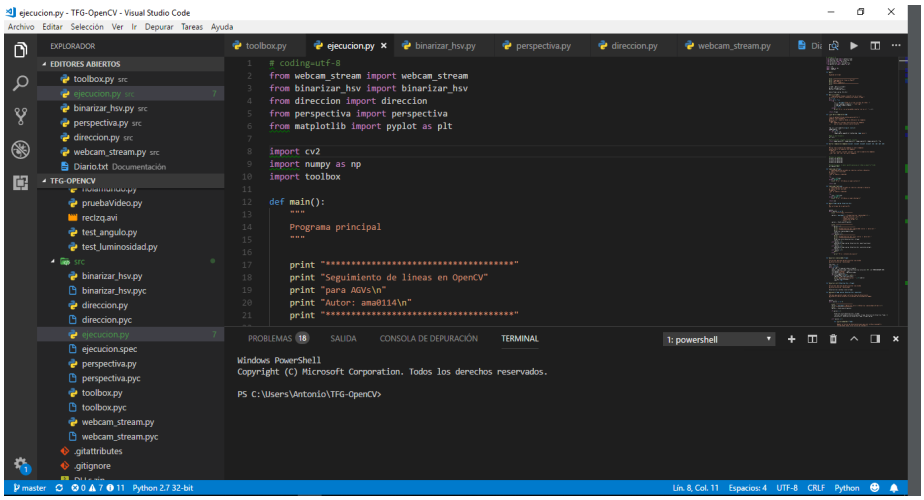

Figura D.1: Interfaz de Visual Studio Code

Git

Git es el sistema de control de versiones elegido para este proyecto. Windows no lo trae por defecto, a si que lo tenemos que descargar desde:

https://git-scm.com/

Descargamos el ejecutable y lo instalamos. Podemos elegir entre usar Git desde una consola Bash, desde la consola de Windows, y también instalar componentes adicionales, esto lo dejamos a gusto del usuario.

Para obtener el proyecto usaremos el comando: git clone https://github.com/ama0114/TFG OpenCV.git

GitKraken

Para gestionar mejor el respositorio, hemos usado la herramienta Git-Kraken. Lo podemos descargar aquí:

https://www.gitkraken.com/

Descargamos el ejecutable y lo instalamos. Esta herramienta también nos permite clonar el repositorio si no lo hemos hecho antes, para ello pinchamos en: File, Clone Repo. Nos pedirá la dirección del archivo git del repositorio: https://github.com/ama0114/TFG-OpenCV.git

En caso de tenerlo ya clonado, pinchamos en la opción: File, Open Repo y buscamos la carpeta donde lo hayamos descargado.

Una vez hecho esto, la herramienta nos mostrará las ramas del proyecto, los commits realizados, en general toda la información del repositorio. También nos permite hacer las operaciones básicas del repositorio, Pull, Push, Stash, Branch y Pop.

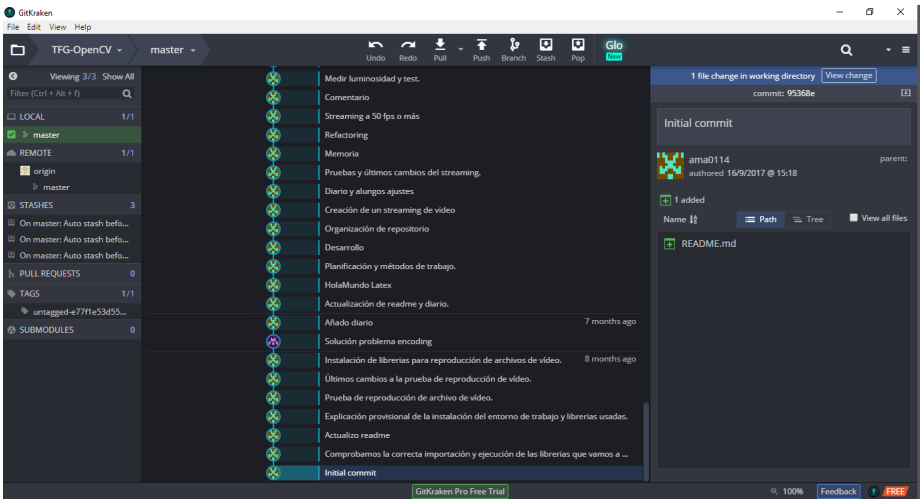

Figura D.2: Interfaz de GitKraken

D.4. Compilación, instalación y ejecución del proyecto

Lo primero que tenemos que hacer es iniciar nuestro servidor de vídeo. Para este proyecto hemos usado la aplicación android IPWebcam. La podemos descargar desde aquí: https://play.google.com/store/apps/details?id=com.pas.webcam&hl=en

El streaming se ha de configurar en una resolución de 176x144. La calidad del mismo se puede ajustar ahora desde la aplicación, o más adelante desde la web.

Asegurarnos que el ordenador donde vamos a ejecutar el proyecto y la aplicación tienen conexión. Para ello ir a la dirección IP que nos muestra la aplicación en la pantalla del smartphone en cualquier navegador de Internet en el ordenador.

Figura D.3: Pantalla de la aplicación IPWWebcam en el smartphone.

Aquí podemos ver la dirección IP que usa la aplicación. En mi caso: 192.168.1.10:8080

Si tenemos conexión deberíamos de ver esto en el navegador:

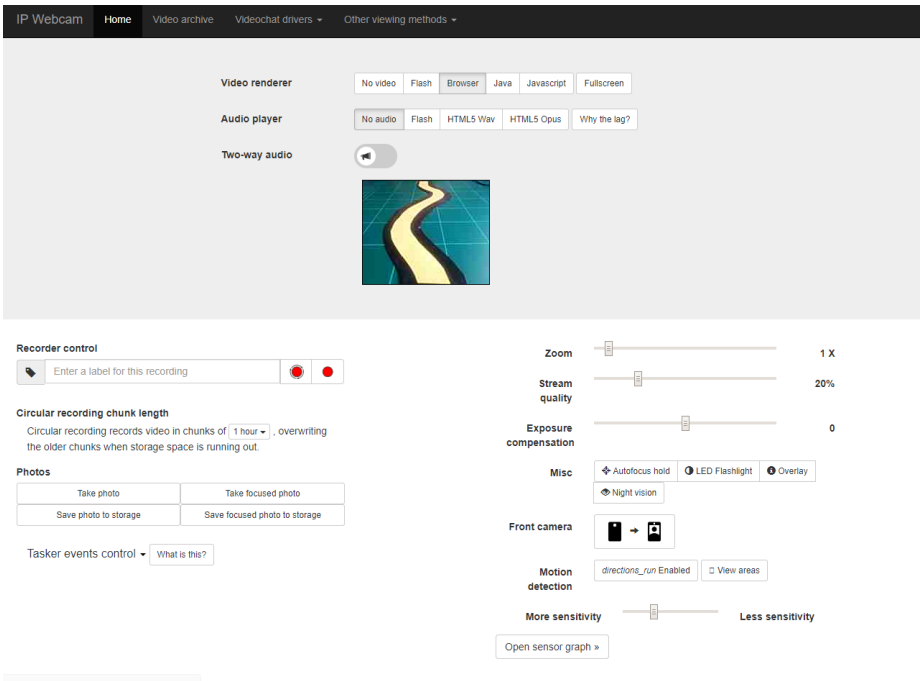

Figura D.4: Servidor web de la aplicación IPWebcam

Desde aquí podemos gestionar el streaming, incluso encender el led de la cámara y enfocar la imagen. También podemos modificar la calidad del streaming.

Cuando el programa solicite la dirección del servidor, hay que dársela de esta forma: http://192.168.1.10:8080 . Basta con hacer copia y pega de la barra del navegador de Internet.

Podemos ejecutar el proyecto de diferentes maneras, veamos:

Linea de comandos

Usando Python desde la linea de comandos, simplemente nos colocamos sobre la carpeta src y ejecutamos el comando:

- python ejecucion.py

El programa se ejecutará.

Visual Studio Code

Podemos ejecutar el programa desde Visual Studio Code.

Hacemos click derecho sobre el archivo ejecucion.py y seleccionamos la opción: Ejecutar archivo Python en la terminal.

Ejecutará el archivo en la terminal del IDE.

Creación del ejecutable

Para crear el ejecutable usaremos la herramienta pyinstaller.

Abrimos una consola de comandos nos movemos con el comando cd hasta la carpeta src y ejecutamos el comando:

- pyinstaller ejecucion.spec

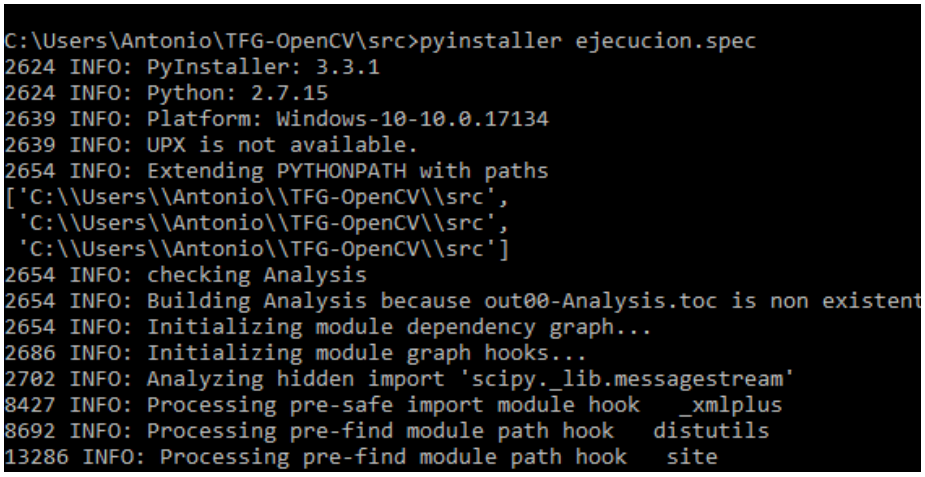

Figura D.5: Inicio de la construcción del ejecutable.

Durante la construcción se generarán varias pantallas de información en la consola de comandos. Acabará en algo igual o similar a esto:

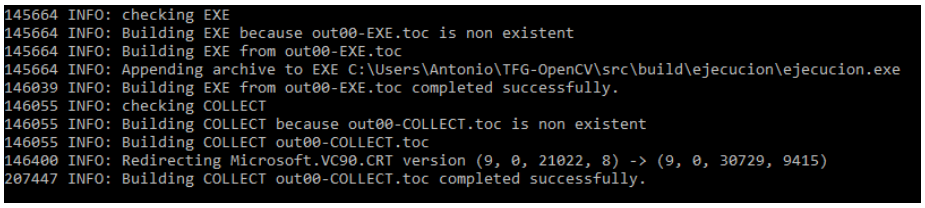

Figura D.6: Fin de la construcción del ejecutable

Si nos lanza algún error, será porque nos faltan Dlls en la carpeta Python/Dlls, recalcar que la carpeta a la que hacemos referencia con Python es la de instalación de Python. Estos Dlls los podemos encontrar en la carpeta Python/libs/site-packages. Dentro de aquí tendremos todos los paquetes que tengamos en Python instalados, buscaremos el paquete del que nos falten los Dlls (nos lo indicará pyinstaller) y copiamos sus Dlls en la carpeta Python/Dlls.

Como esta tarea es un poco complicada, y suele llevar horas de exploración por internet, se ha dejado un fichero Dlls.zip en el repositorio con los Dlls necesarios para crear el ejecutable de esta versión.

OJO! estos Dlls son para Windows 10, si lo queremos construir para otro sistema operativo seguramente necesitemos cambiarlos. Si en un futuro se expande el proyecto, es posible que haya que incluir más.

Tendremos nuestra carpeta con el ejecutable y todos los archivos necesarios en la carpeta src/dist.

La carpeta se ha de distribuir así tal cual, sin modificar nada. El ejecutable en concreto es el archivo ejecucion.exe

D.5. Pruebas del sistema

Las pruebas que se han hecho en este proyecto son principalmente pruebas de investigación, de los diferentes algoritmos, midiendo parámetros y probando su eficacia. Las pruebas se han dejado registradas en vídeos en la carpeta Documentacion/Videos.

https://github.com/ama0114/TFG-OpenCV/tree/master/Documentaci%C3%B3n/Videos

Además se ha generado un diario donde se iban recogiendo todos los resultados de las pruebas, el archivo es Documentacion/Diario.txt.

https://github.com/ama0114/TFG-OpenCV/blob/master/Documentaci%C3%B3n/Diario.txt

D.6. Cambiar conexión de vídeo

Si queremos cambiar la forma en la que el programa obtiene el vídeo, simplemente debemos generar una clase python, con la estructura de la clase $webcam_stream.py$.

El método get_frame(self, modo) se usa para obtener un único frame, como hacer una foto. El parámetro modo indica el formato de la imagen, 0 para blanco y negro, 1 para RGB.

Si observamos la implementación, vemos que hacemos un bucle de 10 iteraciones para que el buffer de donde se obtiene el vídeo, tenga algún fotograma, sin este bucle, la primera imagen devuelta sería gris, porque el buffer está vacío, y hay que hacer una serie de peticiones para que al menos tenga un frame.

El método get_video_stream(self, modo) se usa para obtener frames para realizar una reproducción de vídeo. El parámetro modo indica el formato de la imagen, 0 para blanco y negro, 1 para RGB.

En este caso, no tiene bucles internos, porque se ha de usar dentro de un bucle, el bucle donde querremos obtener el vídeo. Si usamos esta función fuera de un bucle, nos devolverá una imagen gris, puesto que con una sola llamada a la función que conecta con la cámara, no es capaz de llenar el buffer a tiempo.

Si queremos generar una clase con una forma distinta de obtener el vídeo, simplemente tenemos que cumplir la funcionalidad de estos métodos.

get_frame(self, modo) tiene que devolver un único frame, se usa para obtener fotos.

get_video_stream(self, modo) tiene que poderse usar dentro de bucles para obtener vídeo.

D.7. Utilizar módulos en otras aplicaciones.

Los módulos de este programa se han pensado para ser independientes.

Apéndice E

Documentación de usuario

E.1. Introducción

En este apartado veremos como usar la aplicación.

E.2. Requisitos de usuarios

Los requisitos para poder usar la herramienta son:

- Un smartphone con la aplicación IPWebcam, generando un servidor de vídeo accesible en red local a través de la dirección IP mostrada en la pantalla de la aplicación.
- Un ordenador con Windows 10 donde ejecutar el programa, con conexión a la red local donde este el servidor de vídeo.
- Unas plantillas para probar y ejecutar las diferentes funcionalidades. En concreto: una plantilla de un cuadrado negro sobre fondo blanco, una plantilla de un cuadrado de algún color llamativo con borde negro, una línea guía negra sobre fondo blanco, una línea guía de algún color llamativo con borde negro.

E.3. Instalación

La herramienta en sí no tiene instalación, simplemente nos la tenemos que descargar desde la release del repositorio.

https://github.com/ama0114/TFG-OpenCV/releases/tag/untagged-e77f1e53d55c0df5d337

- Descargamos el fichero release_v1.0.zip
- Dentro de este fichero está la carpeta bin, la extraemos donde queramos.

Y ya está, hecho esto ya tendremos la herramienta lista para funcionar en nuestro sistema.

E.4. Manual del usuario

Inicialización

Lo primero que nos solicitará la herramienta es la dirección IP de nuestro servidor de vídeo. Se la proporcionaremos con el formato: http://direccionIP:Puerto

Por ejemplo: http://192.168.1.10:8080

Una vez hecho esto, el programa comprobará que tenemos conexión, si no tenemos conexión, nos dará error y nos volverá a pedir la dirección.

Si tenemos conexión, nos llevará al menú principal:

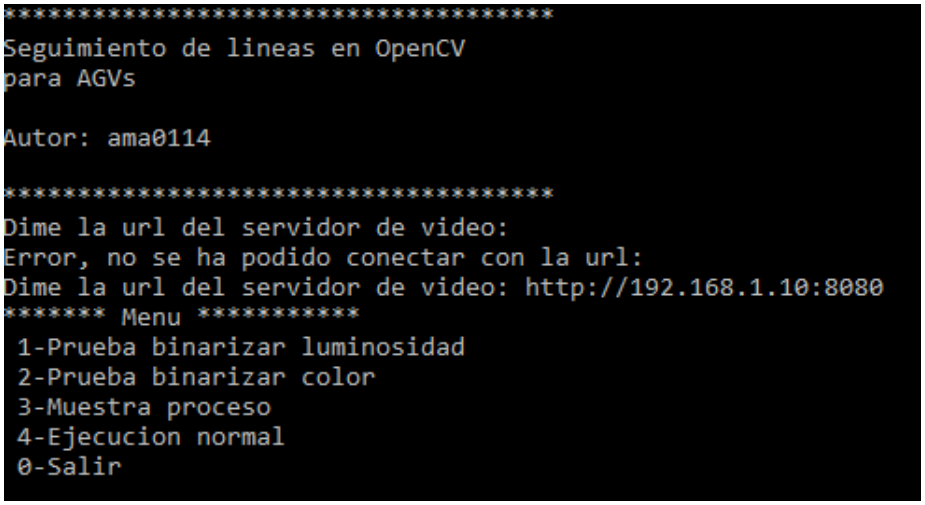

Figura E.1: Inicio de la aplicación

Una vez en el menú principal tenemos 5 opciones:

- 1-Prueba binarizar luminosidad
- 2-Prueba binarizar color

- 3-Muestra proceso
- 4-Ejecucion normal
- 0-Salir

Elegiremos una de las 5 opciones escribiendo el número asociado en la consola de comandos.

Veamos cada una de ellas:

1-Prueba binarizar luminosidad

Esta funcionalidad está pensada para que el usuario pueda probar diferentes materiales y entornos para comprobar la binarización por luminosidad. Al elegir esta opción con el valor 1 se nos abrirá una ventana de OpenCV como esta:

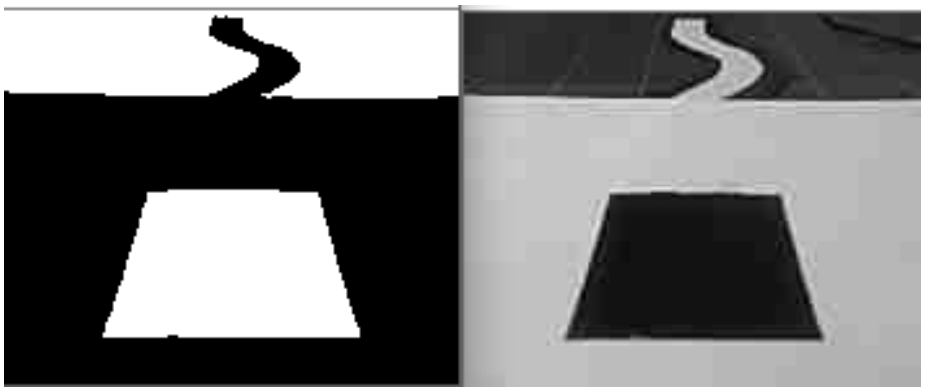

Figura E.2: Ventanas de OpenCV eligiendo la opción 1 del menú principal.

Vemos dos imágenes, una binaria y otra en blanco y negro. Para realizar correctamente la binarización, tenemos que colocar elementos negros sobre fondo blanco. En la imagen anterior podemos ver la binarización de la plantilla del cuadrado negro sobre fondo blanco.

Para salir pulsamos S en el teclado cuando estamos sobre cualquiera de las ventanas.

Nos imprimirá los Fps conseguidos.

Volveremos al menú principal.

2-Prueba binarizar color

Esta funcionalidad está pensada para que el usuario pueda probar diferentes materiales y entornos para comprobar la binarización por color. Al elegir esta opción con el valor 2 se nos abrirá una ventana de OpenCV como esta:

Figura E.3: Ventanas de OpenCV eligiendo la opción 2 del menú principal.

Vemos dos imágenes, una binaria y otra el color. Para seleccionar el color que queramos, pinchamos sobre la imagen en color. Por ejemplo, si queremos el azul, click sobre el azul:

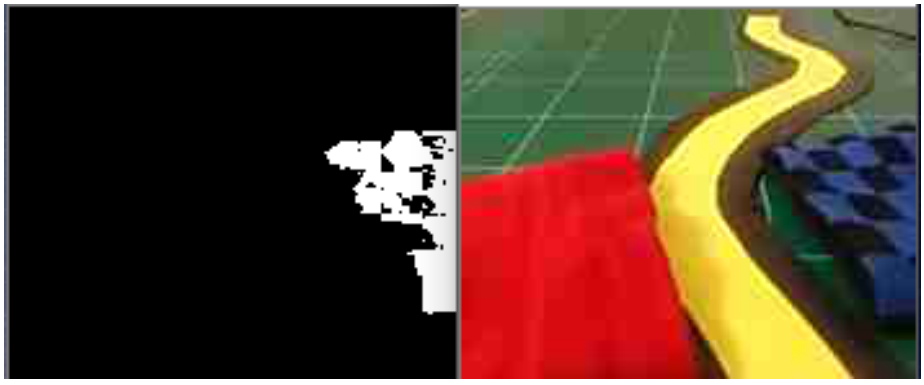

Figura E.4: Binarización si elegimos el color azul.

Para salir pulsamos S en el teclado cuando estemos sobre cualquiera de estas ventanas.

Volveremos al menú principal.

3-Muestra proceso

Esta funcionalidad está pensada para que el usuario pueda probar como funciona la corrección de distorsión de perspectiva, y detectar posibles anomalías en la imagen corregida. Además se mostrará como se detecta la trayectoria. Para entrar en esta opción escribiremos un 3 en la línea de comandos.

Lo primero que nos pedirá es que elijamos el método de binarización que queramos, en un submenú con estas opciones:

- 1-Binarizar por color
- 2-Binarizar por luminosidad
- 0-Salir

La opción de salir nos devolverá al menú principal.

Si elegimos binarización por color se nos abrirá una ventana como la Figura E.3. Para calibrar el color seguiremos los mismos pasos de la Prueba de binarizar por color.

Si elegimos binarización por luminosidad, pasaremos al siguiente menú.

En el siguiente menú se nos pedirá calibrar la distorsión de perspectiva. Nos aparecerá un mensaje en la línea de comandos como este: Pulsa intro para calcular el coeficiente de corrección de distorsión por perspectiva, pulsaremos intro y se nos abrirá una ventana con la imagen binarizada.

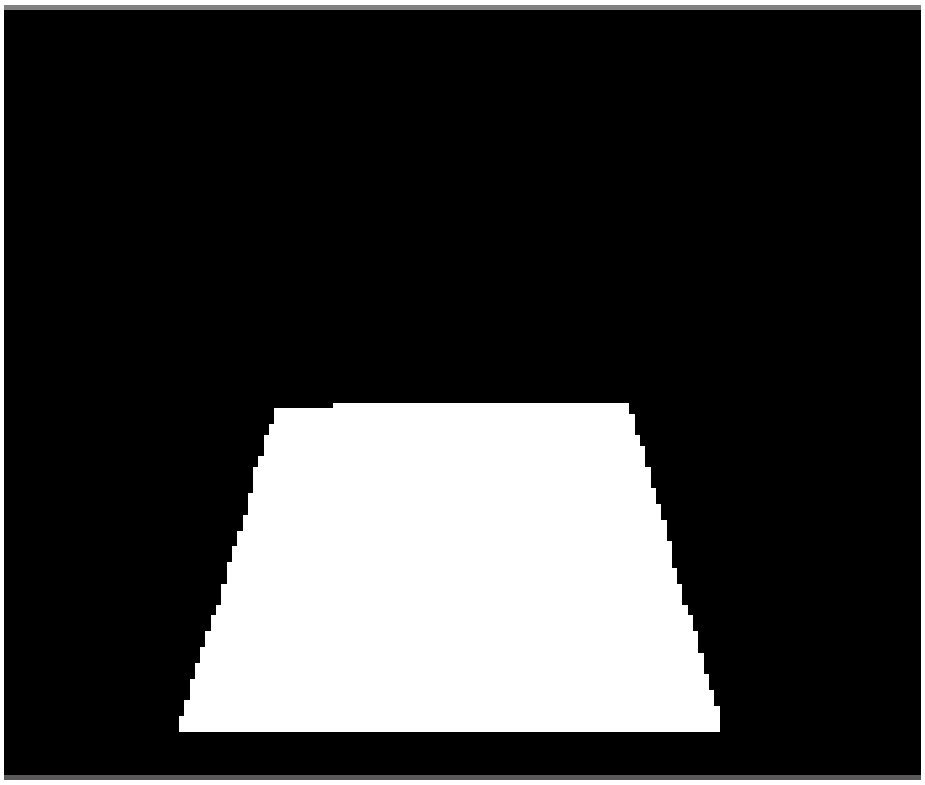

Figura E.5: Imagen de calibración de la distorsión de perspectiva

Para que la corrección sea adecuada, debemos colocar el cuadrado tal cual aparece en la imagen anterior, buscando que este un poco por encima del limite inferior de la imagen, y lo más recto posible, si es posible que forme una línea recta perfecta en la parte inferior y superior, mejor.

Una vez hecho esto, pulsamos S para salir de esa ventana, y en la ventana de comandos nos aparecerá el coeficiente calculado de la distorsión de perspectiva.

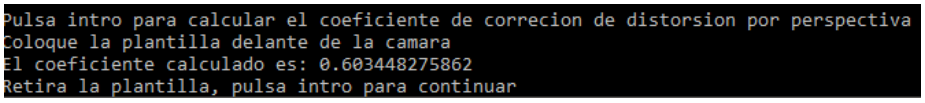

Figura E.6: Línea de comandos tras corregir distorsión de perspectiva

El coeficiente está relacionado con el ángulo que forma la cámara con el suelo, podemos consultar la relación en la Tabla E.1.

Angulo	Coeficiente
40º	0.506
45º	0.530
50º	0.550
55º	0.576
60º	0.613

Tabla E.1: Valores obtenidos en las pruebas de corrección de distorsión de perspectiva

Pulsamos enter para continuar. A continuación se abrirá una ventana como esta:

Para binarización por color:

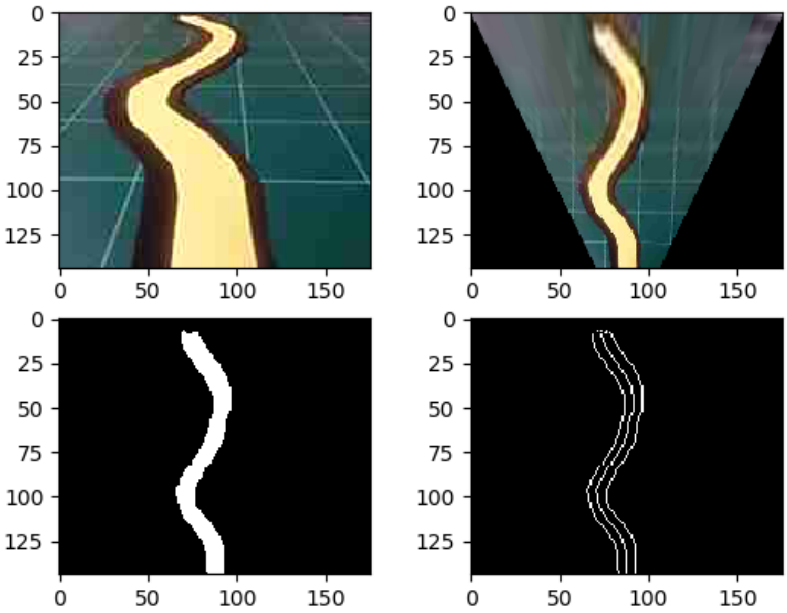

Figura E.7: Ventana de comparación de imágenes en binarización por color

Para binarización por luminosidad:

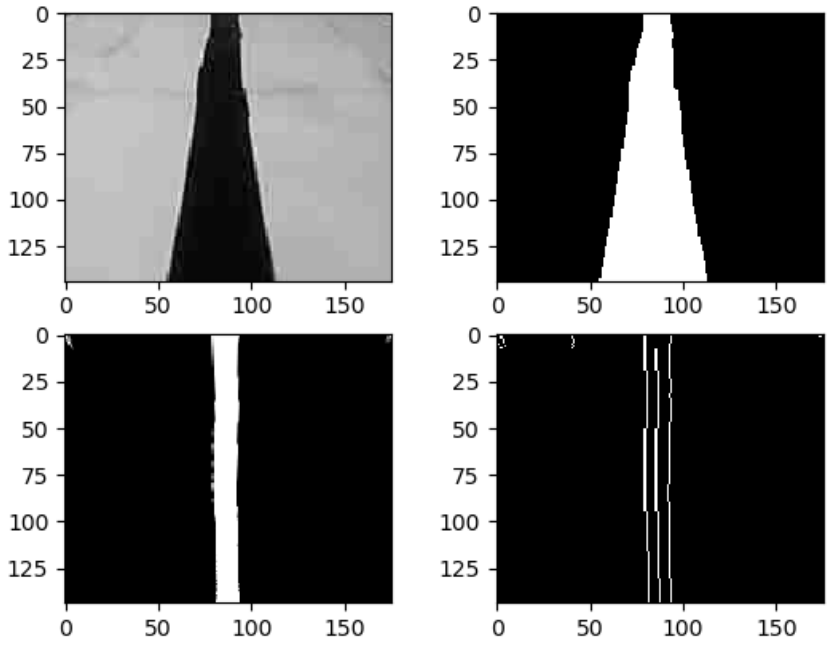

Figura E.8: Ventana de comparación de imágenes en binarización por color

Podemos ver que las imágenes que aparecen entre la binarización por color y por luminosidad son diferentes. Esto se debe a que al binarizar por luminisdad, si o si hay que hacer siempre primero la binarización y luego la correción de perspectiva, ya que la correción de perspectiva genera dos zonas grandes negras que después son interpretadas como objetos por la binarización, lo cual es erroneo. En cambio en la binarización por color es indiferente realizar primero la binarización o la corrección.

En esta ventana el usuario puede ver la imagen real, la imagen binarizada, la imagen corregida, en vista de pajaro, y la detección de la trayectoria.

Para salir simplemente cerramos con el aspa roja esta ventana. Nos devolverá al submenú para elegir el tipo de binarización, si queremos volver al principal, escribiremos un 0 en la línea de comandos.

4-Ejecución Normal

Esta opción permite ejecutar el flujo principal del programa, donde se usarán los sistemas de guiado y de detección de trayectoria, además de algunas otras funcionalidades que iremos viendo.

Entraremos escribiendo un 4 en la línea de comandos desde el menú principal.

Nos pedirá elegir el tipo de binarización que queramos usar, realizar la calibración si es por color, y realizar la calibración de perspectiva. Todo esto es similar a los pasos realizados en la opción 3-Muestra Proceso.

A continuación nos pedirá el valor para el rango seguro de la dirección.

Veamos que es esto:

El sistema de guiado funciona detectando la distancia de la trayectoria al centro de la imagen, si está en el centro, tenemos que seguir recto, si está a la izquierda o a la derecha, tendremos que girar hacia ella, para volver a estar centrados.

Lo que permite hacer el rango seguro es crear una "zona.ªl rededor del centro de la imagen, en la que también el programa nos mande ir recto, para que no este continuamente corrigiendonos. Esto se basa en la propia conducción humana, donde no estamos buscando el centro ideal del carril si no que tenemos unos margenes por donde movernos, y cuando nos acercamos a uno de ellos, corregimos la trayectoria.

En esta opción se nos pedirá un valor entre 0 y 0.3. Esto se multiplicará por el tamaño de la imagen en horizontal, y generará un valor en pixels, ese valor será la distancia entre el centro y los bordes del rango seguro donde el programa nos seguirá diciendo que continuemos recto.

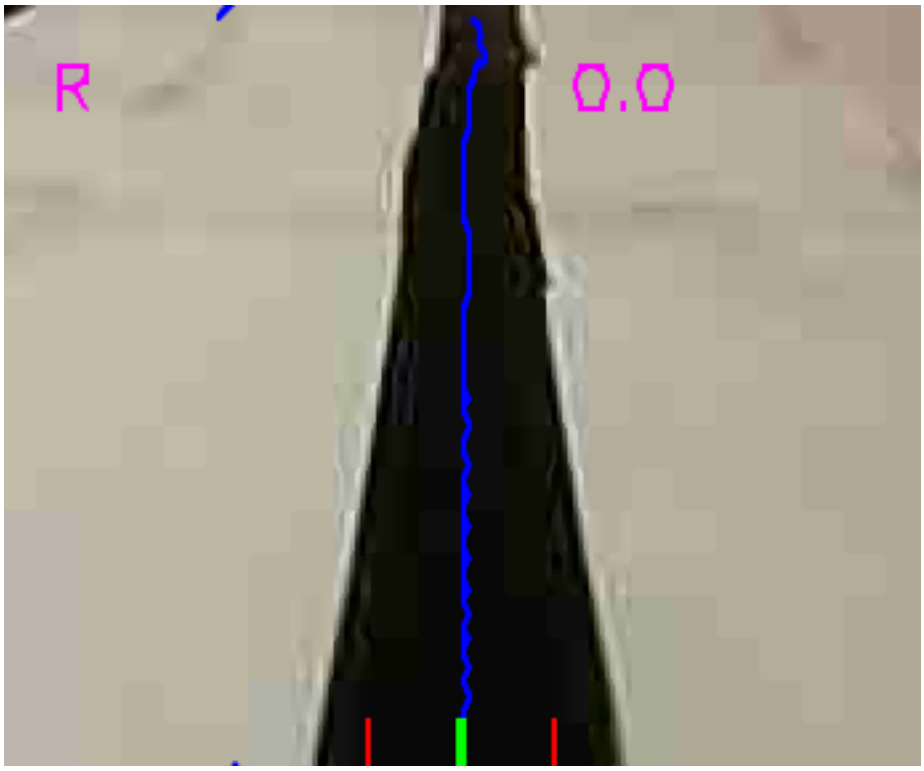

Figura E.9: Imagen donde se aprecian los márgenes del rango seguro de la dirección

En esta imagen podemos ver en la parte inferior en rojo los bordes del rango seguro, en verde está marcado el centro de la imagen.

Una vez hayamos introducido un valor, nos pedirá el ángulo de giro de nuestro vehículo. En este caso como no trabajamos con el vehículo, puede ser cualquiera, pero ha de ser entero positivo. Esto permitirá conocer al programa el ángulo máximo de giro que tiene el vehículo.

```
Dime el rango seguro para la direccion
 min=0, max=0.3
0.2
Dime el angulo de giro maximo del vehiculo[Valor entero]: 90
```

Figura E.10: Ajuste del rango seguro y ángulo de giro de la dirección

Una vez hecho esto, se nos abrirá una ventana como esta:

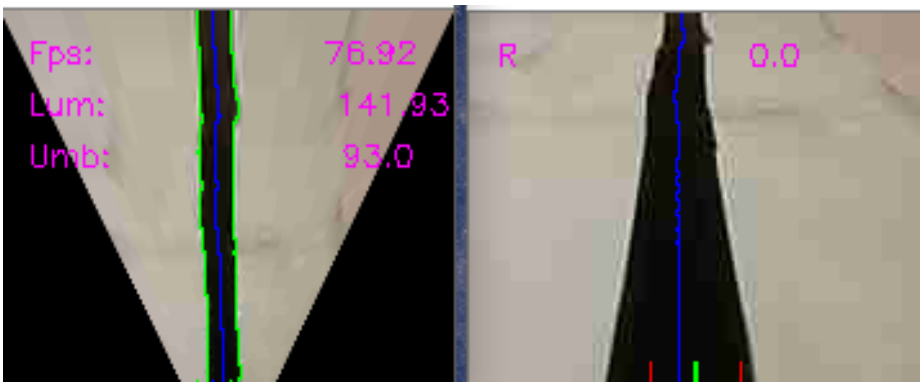

Figura E.11: Ventana del sistema de guiado y calculo de la trayectoria en perspectiva, binarización por luminosidad

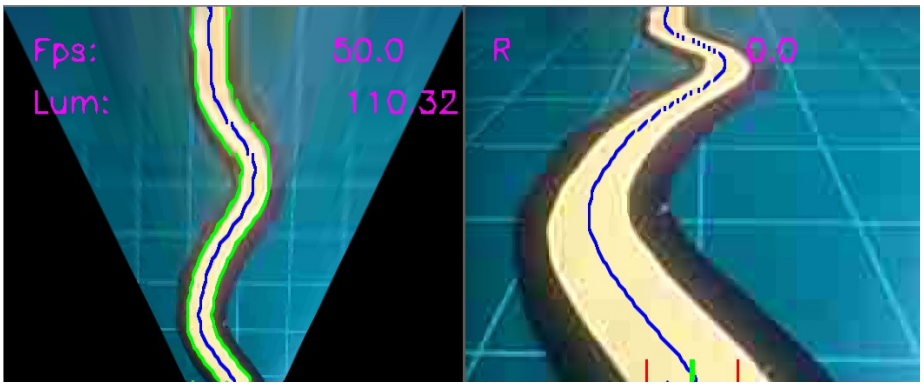

Figura E.12: Ventana del sistema de guiado y calculo de la trayectoria en perspectiva, binarización por color

Vamos a explicar los diferentes elementos.

En la parte izquierda tenemos la distorsión por perspectiva resuelta, así como la detección de la trayectoria. También podemos ver 3 valores numéricos, los Fps a los que está funcionando el programa, la luminosidad (Lum) ambiente, y el umbral(Umb) de binarización de Otsu, este solo aparece si la binarización es por luminosidad.

Estos valores nos darán información sobre el sistema y la situación del entorno en el que estamos trabajando.

En la parte derecha tenemos el sistema de guiado. Vemos un indicador textual y uno numérico, que pueden tomar estos valores en función de la posición de la trayectoria respecto del centro de la imagen.

Para el indicador textual:

- RI, recto izquierda, indica que el ángulo se tiene que interpretar como un giro a la izquierda. Para ángulos menores o iguales de 45º.
- RD, recto derecha, indica que el ángulo se tiene que interpretar como un giro a la derecha. Para ángulos menores o iguales de 45º.
- I, similar a RI pero para ángulos mayores de 45º.
- D, similar a RD pero para ángulos mayores de 45º.

El indicador numérico nos dirá los grados de giro, asociados a la dirección del indicador textual.

Podemos ver como el ángulo de giro disminuye según nos acercamos a la línea, para realizar correcciones de dirección lo más suaves posible.

Es importante destacar que si no vemos uno de los bordes de la línea en la parte inferior de la imagen, el sistema no podrá encontrar la trayectoria y dará error.

Cuando nos salgamos de los margenes de la zona segura, el margen por el que nos hayamos salido se moverá al centro, dando lugar a la activación del sistema de guiado, entonces veremos como la R cambiará a RI o RD, y el ángulo pasará de ser 0 a tener un valor. Ese valor son los grados que debemos de girar la dirección en el sentido RD o RI, para volver a centrarnos sobre la línea guía.

Para salir de esta opción, pulsaremos S en el teclado. Nos devolverá al menú donde se nos permite elegir el tipo de binarización, si queremos salir, escribimos un 0 en la linea de comandos y pulsamos enter.

Bibliografía

[1] Antonio de los Mozos Alonso. Diario de pruebas, 2018. [Internet; consultado 27-junio-2018] https://github.com/ama0114/TFG-OpenCV/blob/master/Documentaci%C3%B3n/Diario.txt.

[2] Philipp. What are the essential differences between bsd and mit licences?, 2015. [Internet; consultado 27-junio-2018] https://opensource.stackexchange.com/questions/217/what-are-the-essential-differences-between-bsd-and-mit-licences.

[3] Python. Choosing a license, 2018. [Internet; consultado 27-junio-2018] http://docs.python-guide.org/en/latest/writing/license/.

[4] Notario Francisco Rosales. Robots e inteligencia artificial, concepto y normativa, 2016. [Internet; consultado 27-junio-2018] https://www.notariofranciscorosales.com/robots-e-inteligencia-artificial-concepto-y-normativa/.

[5] Seguridad Social. Bases y tipos de cotización 2018, 2018. [Internet; consultado 27-junio-2018] http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm.

[6] Wikipedia. Scrum (desarrollo de software), 2018. [Internet; consultado 27-junio-2018] https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software).

[7] Wikipedia. Software release life cycle, 2018. [Internet; consultado 27-junio-2018] https://en.wikipedia.org/wiki/Software_release_life_cycle.