Detector de senyals de trànsit informatives en temps real

Guillem Campo Fons

Resum– El propòsit d'aquest projecte consisteix en el desenvolupament d'una eina de reconeixement de senyals de trànsit a temps real, mitjançant visió per computador. El sistema pot reconèixer les diferents senyals de trànsit agrupades en tres categories basades en les existents de la *'Dirección General de Tráfico' (DGT)*. Aquestes categories son les senyals d'orientació blaves, les senyals d'orientació blanques i les senyals d'indicació general. Un cop detectades i classificades aquestes senyals seran mostrades al usuari en funció del tipus i l'ordre en que el sistema les ha reconegut.

Paraules clau– Reconeixement, DGT, visió per computador, orientació, indicació, classificades

Abstract– The purpose of this project consists on the implementation of a recognising tool of traffic signals in real time, making use of computer vision. The system can recognise different traffic signals grouped in three categories based on the existing ones defined by the *'Dirección General de Tráfico' (DGT)*. Those categories are the blue orientation signals,white orientation signals, and general indication signals. Once detected and classified, those signals will be showed to the user depending on the type and order where the system recognise them.

Keywords– Recognising, DGT, computer vision, orientation, indication, classified

1 Introducció

ACTUALMENT s'han produït una sèrie d'avenços tecnològics els quals ens han permès apropar-nos al fet en que una màquina pugui arribar a percebre la realitat tal i com ho faria una persona. En aquest cas parlem de la visió per computador [1], la qual mitjançant un conjunt d'eines i mètodes que permeten obtenir, processar i analitzar imatges del món real, amb la finalitat de poder ser tractades per un ordinador. Això ens permet automatitzar una amplia gamma de tasques al aportar a les màquines la informació que necessiten per prendre decisions i executar-les.

L'objectiu principal que es vol assolir en aquest projecte es el de reconèixer senyals de trànsit informatives en temps real fent ús d'aquesta visió per computador, i mostrar-les als usuaris a través d'una pantalla. Aquest objectiu es podria derivar a moltes funcionalitats com son un sistema de guia per al conductor d'un automòbil, o inclús que el propi automòbil pugui actuar en funció de les senyals que reconeix com un vehicle de conducció autònoma [2].

Els objectius finals del projecte, es centraran en:

- La gravació i extracció de frames a temps real mitjançant l'ús d'una càmera.
- El reconeixement de senyals de trànsit (Orientació i indicació).
- La classificació de les senyals de trànsit detectades.
- Mostrar els resultats obtinguts en una interfaç d'usuari.

D'aquesta manera si el conductor ha deixat una senyal enrere que no ha vist o no li ha donat temps a llegir la pot veure en una pantalla.

En aquest document tractarem la metodologia utilitzada, els algorismes utilitzats, el tractament de les imatges, el reconeixement de senyals, etc. Tot seguit s'exposaràn els resultats obtinguts i finalment s'extreuràn unes conclusions i les possibles línies de continuació d'aquest projecte.

2 Planificació del projecte

Per tal d'assolir el correcte funcionament d'aquest projecte s'ha seguit la planificació següent al llarg del seu desenvo-

• E-mail de contacte: guillem.campo@e-campus.uab.cat
• Menció realitzada: Computació
• Treball tutoritzat per: Jordi Serra (DDC)
• Curs 2020/21

lupament. Dividirem el projecte en tres fases de cinc setmanes:

1. La primera fase del projecte ha d'assolir la gravació d'un vídeo a temps real, l'extracció de frames cada x segons d'aquest vídeo i l'inici de la implementació de la detecció de senyals.
   (a) Gravació a temps real mitjançant l'ús d'una càmera. *(1a setmana)*
   (b) Extracció de frames sobre un vídeo a temps real. *(1a setmana)*
   (c) Inici de la implementació de detecció de senyals sobre els frames extrets. *(2a, 3a, i 4a setmana)*
   (d) La última setmana es reserva per a la recopilació de resultats obtinguts, analitzar el correcte funcionament del projecte i afegir possibles millores. *(5a setmana)*
2. La segona fase del projecte consistirà en fer que sigui capaç de poder separar les senyals detectades dels frames, i començar a fer la classificació d'aquestes per posteriorment mostrar-les per pantalla.
   (a) Identificació de senyals. *(1a i 2a setmana)*
   (b) Inici de la classificació de senyals extretes del frame. *(3a i 4a setmana)*
   (c) La última setmana es reserva per a la recopilació de resultats obtinguts, analitzar el correcte funcionament del projecte i afegir possibles millores. *(5a setmana)*
3. La tercera i última fase ens servirà per acabar de realitzar el procés de classificació de les senyals i mostrarles per pantalla.
   (a) Classificació de les senyals segons el tipus. *(1a setmana)*
   (b) Mostrar per pantalla les senyals classificades. *(2a setmana)*
   (c) La tercera i quarta setmana es reserva per a la recopilació de resultats obtinguts, analitzar el correcte funcionament del projecte i afegir possibles millores. *(3a i 4a setmana)*
   (d) Polir el rendiment i disseny del codi. *(5a setmana)*

3 METODOLOGIA

Un cop feta la planificació el pròxim pas es quina metodologia utilitzar per a realitzar el desenvolupament del projecte.

3.1 Eines utilitzades

Per tal de desenvolupar el projecte s'ha fet ús del llenguatge Python [3] i d'una sèrie d'eines, entre les quals tenim:

- La llibreria *OpenCV (Open Source Computer Vision)* [4], la qual utilitzarem per a la major part de tasques de visió per computador per a la realització del projecte.
- Les llibreries *numpy* [5], *pathlib* [6] , *os* [7] , i *glob* [8] per a taques menors però importants en el desenvolupament del projecte, com poden ser lectura de fitxers, tractament d'arrays, etc.
- Les llibreries *tkinter* [9] i *PIL* [10] per a mostrar els resultats finals en una 'Graphic user interface' (GUI) [11] .
- La llibreria *pytesseract* [12] la qual ens permetrà aplicar un contenidor python de Google Tesseract-OCR (Optical Character Recognition) per tal de classificar millor les senyals.
- La llibreria *multiprocessing* [13] per treballar amb varis processos simultàniament.

Conforme es vagin explicant els diferents passos seguits en la metodologia s'aniran veient els usos d'aquestes eines mencionades.

3.2 Gravació i extracció de frames

El primer pas per a la realització del projecte es tractarà de la gravació d'un vídeo a temps real, això ho podem aconseguir gràcies a la llibreria *openCV* la qual ens proporciona una funció per aquesta tasca anomenada *VideoCapture* [14]. Tot seguit a partir d'aquest vídeo generat extreurem els frames cada x segons, en el nostre cas s'ha sel·leccionat que extregui frames cada 5 segons , però aquest valor el podem variar. Un cop extrets aquests frames, es guarden en una carpeta de la qual seran extrets per tal de processar-los, i d'aquesta manera trobar les senyals de trànsit. En la Fig.1 podem apreciar com es veuria un frame extret del video.

Fig. 1: Frame extret del vídeo

Un cop el frame es guardat en la carpeta, aquest es agafat i es preprocessat de manera que es redueix el soroll de la imatge amb un mètode de la llibreria *openCV* el qual aplica l'algoritme *Non-local Means Denoising* [15] i se li aplica un altre mètode anomenat *median Blur* [16] per tal de que a l'hora que es redueix encara mes el soroll aconseguim eliminar colors que tenen poca repercusió en la imatge, de manera que ens queda una imatge com la Fig.2

Un cop s'ha reduït el soroll del frame es vol definir una 'ROI' (Region of Interest) [17] , tal i com es veu en la Fig.3, per tal de que la detecció de senyals sigui en una zona específica, ja que d'aquesta manera evitarem que es puguin detectar objectes innecessaris com poden ser cotxes a la carretera o altres objectes.

Fig. 2: Frame netejat de soroll

Fig. 3: Regió d'Interés dels frames

3.3 Detecció i classificació de senyals

Ara que ja tenim la regió d'interès del frame definida, el pròxim pas es convertir la imatge *RGB (Red, Green, Blue)* [18] en una imatge *HSV (Hue, Saturation, Value)* [19], per tal de poder definir uns valors per a la detecció de colors, ja que mitjançant *HSV* podem objetnir colors diferents en funció de la sseva il·luminació i saturació, de manera que la detecció de colors es més precisa que en *RGB*, tal i com podem apreciar en la Fig.4 . Els colors que volem trobar vindran definits per un array de tres valors en ordre *BGR (Blue, Green, Red)*, els quals s'hauran de detectar en la imatge *HSV*.

Fig. 4: Frame convertit a HSV

Per tal de trobar els colors es defineix un valor de color *BGR*, però com que no podem manipular la il·luminació dels frames, el que es fa es agafar varis rangs per a un color de manera que tots els colors que estiguin en aquell rang seran detectats com els que podem apreciar en la Taula 1, així un color amb diferents tonalitats podrà ser trobat. En aquest cas s'han buscat colors *HSV* per blaus i colors *HSV* per blancs. Aquests rangs de colors han estat establerts gracies a un programa independent de l'original, que hem creat, el qual proporcionant-li una imatge, aquest t'obre una interfaç en la qual pots anar cambiant els valors per tal de trobar colors en la imatge HSV, com podem veure en la Fig.5.

Per tal de processar encara més la imatge i poder trobar més varietat de colors en funció de la situació, primer es fa servir un rang petit per a que trobi les senyals, i si no en troba cap a causa de qualsevol imprevist aquest rang s'amplia, així ens estalviem que trobi objectes innecessaris en la primera cerca de colors, i en la segona encara que trobi objectes innecesaris podran ser filtrats més endavant juntament amb les senyals detectades.

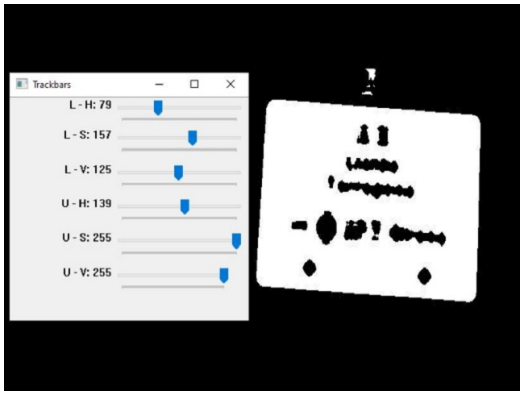

Fig. 5: Interfaç per trobar colors en la imatge HSV

TAULA 1: RANGS DE COLORS DESITJATS

1r Rang blau
BGR = 78 , 158 , 124
BGR = 138 , 255 , 255
2n Rang blau
BGR = 106 , 80 , 2
BGR = 130 , 255 , 255
1r Rang blanc
BGR = 0 , 0 , 200
BGR = 91 , 37 , 204
2n Rang blanc
BGR = 74 , 0 , 160
BGR = 91 , 37 , 204

Ara que ja tenim els rangs per cada color s'han de crear mascares d'aquests, de manera que tot color que estigui dins aquest rang sigui extret en una màscara. Per tal de visualitzar aquesta màscara generarem un frame on es vegi només el color que volem, de manera que agafarem la imatge original i li aplicarem la màscara, donant com a resultat una imatge en la que només es veurà el color que es vol aconseguir. Tot seguit es convertirà aquesta imatge en una imatge en escala de grisos i se li aplicarà un *threshold* [20] per tal d'aïllar encara més el resultat desitjat. Un cop tenim aquesta imatge amb el *threshold* aplicat, eliminem el soroll de la imatge i ja tenim l'objecte aïllat per al color buscat, com es pot contemplar en la Fig.6.

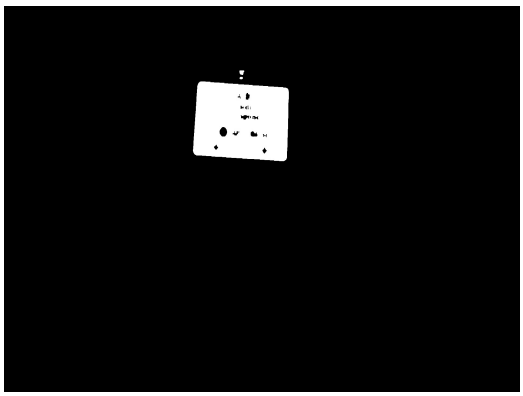

Fig. 6: Objecte desitjat aïllat del frame

Es seguirà el mateix procediment per a cada color que vulguem de manera que obtindrem una imatge per color buscat.

Arribats a aquest punt, es el moment de trobar els contorns dels objectes. Gràcies a la llibreria *openCV* tenim un mètode que ens facilita molt la feina, el qual s'anomena *findContours* [21], aquest ens trobara el contorn més extern dels objectes de la imatge que hem generat anteriorment com els de la Fig.7, i generarà una llista de contorns.

Fig. 7: Contorns trobats per als colors blau i blanc

Per tal d'escollir el contorn que més s'adequa al que volem aconseguir es realitza un filtratge d'aquests.

El primer filtrat té relació amb la mida dels contorns, si un contorn ocupa més de 3/4 parts de l'amplada o més de la meitat de l'alçada que la pròpia imatge, aquest contorn no serà viable ja que es molt probable que estigui detectant el cel o qualsevol ombra que ocupi tota la imatge. El mateix passa si aquell contorn es més petit de 40*px*, ja que vol dir que l'objecte està molt lluny i per tant no es veurà amb claredat o pot voler dir que es un objecte petit que no ens interessa obtenir. En la Taula 2 es poden apreciar millor aquests filtres de la mida dels contorns.D'aquesta manera obtenim contorns que no estan lluny, no son petits, ni molt grans, és a dir, que obtindrem contorns que s'adeqüin a les nostres necessitats.

TAULA 2: FILTRES DE LES MIDES DELS CONTORNS

Filtre d'amplada
40<ampladacontorn<ampladaframe-(ampladaframe/4)
Filtre d'alçada
40<alçadacontorn<alçadaframe-(alçadaframe/2)

Tots aquells contorns que hagin passat aquest primer filtratge seran retallats de la imatge original, per a poder realitzar un segon filtrat, aquests retalls es realitzaran agafant els punts més externs del contorn i generant un rectangle a partir d'aquests punts. De manera que obtindrem imatges retallades com la Fig.8.

Fig. 8: Interior del contorn retallat del frame

El segon filtrat al que es sotmet la imatge va en relació al espai entre el contorn del objecte i el rectangle generat. Hi ha contorns d'objectes que han passat el primer filtrat els quals son molt irregulars, això ens permet descartar-los ja que les senyals que volem detectar tenen forma geomètrica. Per tant tal i com es veu en la Taula 3, si el contorn del objecte té molt espai entre aquest i el rectangle que hem generat amb els punts exteriors, sabem que no es una senyal. Per ser més exactes, si l'àrea que hi ha entre el contorn i el rectangle es més petita que 1/3 de l'àrea del rectangle aquesta imatge passa el filtre, sinó es descarta.

TAULA 3: FILTRE DE L'ÀREA DEL RECTANGLE

Filtre geomètric
area_no_desitjada<area_rectangle/3

Com es pot apreciar en la Fig.9 l'àrea que hi ha entre el contorn de l'objecte detectat de color blanc i l'àrea del rectangle és més gran que 1/3 del tamany de l'àrea de la imatge, per tant sabem que aquest objecte es irregular i per tant no és una senyal de trànsit.

Fig. 9: Contorn retallat que no compleix les condicions

Arribats a aquest punt, el programa ha de ser capaç de classificar aquestes senyals per no mostrar-ne de repetides. És a dir ha de separar les senyals d'orientació i indicació que

trobi de color blau, i les senyals d'orientació de color blanc. Per tal d'aconseguir això com que segons el tipus de senyal aquesta té unes mides especifiques, es poden classificar en funció d'aquestes tal i com es veu en la Taula 4.

TAULA 4: TIPUS DE SENYALS SEGONS LES MIDES

Senyal d'Orientació Blava
amplada > alçada
Senyal d'Indicació Blava
alçada > amplada
Senyal d'Orientació Blanques
No reben cap tipus de classificació ja que tenim senyals d'orientació verticals i horitzontals.

Un cop classificades aquestes senyals, cada una rebrà un processament diferent en funció del tipus:

- Les senyals d'orientació blaves seran sotmeses a un reconeixement de caràcters, primer convertirem la imatge a escala de grisos i seguidament aplicarem un *Gaussian Blur* [22] per tal de reduir soroll de la imatge, un cop fet aquest pas aplicarem un *threshold* per tal de detectar més fàcilment les lletres del interior de la senyal, aquest *trheshold* anirà variant fins que trobi algun caràcter que es pugui llegir, de manera que començarà amb un threshold alt i aquest anirà reduint-se fins que es trobi algun caràcter. Un cop aïllades aquestes lletres, fent ús de la llibreria *pytesseract* la qual es un *OCR*, analitzarem quin text és detectat. Si el *OCR* detecta text donem per suposat que és una senyal d'orientació i per tant la donarà per vàlida. Podem apreciar com quedaria la imatge a l'hora de detectar caràcters en la Fig.10 i com *pytesseract* els llegeix en la Taula 5. No és un *OCR* completament exacte ja que les imatges no sempre tenen una bona resolució, però s'adequa a les nostres necessitats.

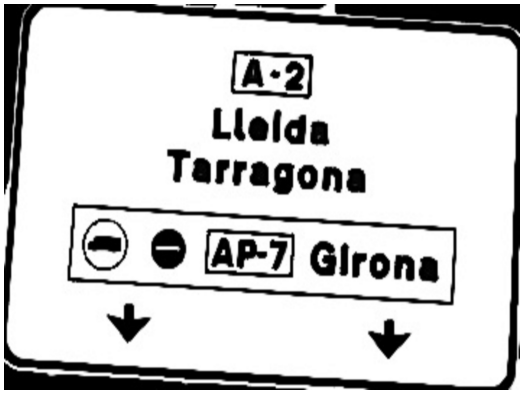

Fig. 10: Imatge que serà sotmesa al reconeixement de caràcters

TAULA 5: TEXT DETECTAT PER *pytesseract* SOBRE LA FIG.10

[a-2 Tarragona es Girona y y

Això ens demostra que *pytesseract* pot no ser completament efectiu per aquests casos.

- Les senyals d'indicació blaves no es sotmetran a cap filtratge addicional ja que arribats a aquest punt s'aproxima a l'objectiu desitjat.
- Les senyals d'orientació blanques patiran el mateix filtrat que les senyals d'orientació blaves,ja que tenen el mateix disseny.

Totes aquestes senyals trobades seran guardades en una carpeta en funció del tipus de senyal que son, i en l'ordre que arriben, tal i com es mostra en la Taula 6. D'aquesta manera tindrem un control per a que no se sobrescriguin unes sobre altres quan hi ha mes d'una detectada en el mateix frame, i per tant en el moment de mostrar-les per pantalla aparegui una de cada tipus.

TAULA 6: FITXERS RESULTATS

Indicació Blaves
0_IndicacioBlue.jpg
1_IndicacioBlue.jpg
...
Orientació Blaves
0_OrientacioBlue.jpg
1_OrientacioBlue.jpg
...
Blanques
0_WhiteOrientacio.jpg
1_WhiteOrientacio.jpg
...

Cal afegir que encara que les senyals blanques es guardin amb el nom WhiteOrientacio, aquestes pertanyen a les d'orientació i indicació.

3.4 Visualització de senyals

Per tant ara que ja tenim tot el procés de detecció i classificació només cal mostrar els resultats per pantalla. Per a la realització d'aquest pas s'ha fet ús de la llibreria *tkinter*, la qual ens permet desenvolupar una interfície gràfica d'usuari *(GUI)* la qual mostrarà les senyals. La *GUI* agafa les senyals guardades en una carpeta on tindrem les senyals d'orientació blaves, indicació i orientació blanques, en funció del nom en que s'han guardat aquestes seran mostrades en una zona concreta de la interfície, i cada cop que entri una senyal nova actualitzarà l'anterior mostrada. De manera que es mostraran com en la Fig.11.

4 RESULTATS OBTINGUTS

En aquesta secció es mostraran els resultats finals obtinguts, tant si son bons com si son erronis, i si son erronis s'explicarà perquè.

Partint d'un vídeo en el qual el cel està ennuvolat, i esta plovent de manera que la càmera agafa frames en que hi ha gotes en el parabrises del cotxe s'han obtingut els resultats com els mostrats en la Fig.12. Es pot apreciar que les senyals es veuen malament a causa de que està plovent i dificulta la visió d'aquestes.

S'han trobat falsos positius com els que es mostren en la Fig.13 i la Fig.14, en un vídeo del que s'han extret 75 frames i han aparegut 5 falsos positius. Aquests es donen per

Fig. 11: Exemple de *GUI* mostrant els resultats

culpa de que la forma que tenen es molt semblant al que volem trobar, i el fet de que continguin text alguns fa creure al codi que realment son senyals d'orientació o indicació, l'altre factor que provoca això es que hi ha formes que l'OCR reconeix com a lletres per tant també enganya al programa per dir-ho d'alguna forma. Un altre aspecte a tenir en compte es que no han aparegut falsos positius blaus, aixo es degut a que el cel en un dia de nuvol es blanc, i això ha donat peu a que com es pot veure en la Fig.13 esta detectant un objecte en el cel blanc.

Com es pot apreciar en la Fig.15, aquesta senyal no ha estat detectada a causa de que hi ha gotes que dificulten la detecció d'aquesta. El codi no es capaç de llegir les lletres. A més la resolució de la imatge no es gaire bona i per tant dificulta encara més aquesta detecció.

Fig. 12: Resultats obtinguts en un dia de pluja

Fig. 13: Fals positiu en un dia de pluja

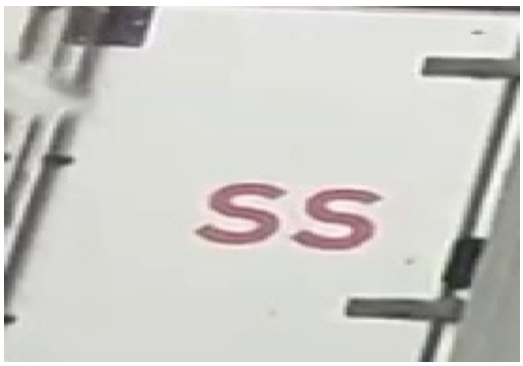

Fig. 14: Fals positiu en un dia de pluja

Fig. 15: Senyal no detectada en un dia de pluja

En l'apèndix mostraré totes les imatges que ha detectat el programa en un dia de pluja.

Per als resultats en que el sol està radiant, i el cel està aclarit sense cap núvol, s'han fet servir frames independents que es trobessin en aquestes condicions, per tal de comprovar el seu funcionament. Podem apreciar aquests resultats en les Fig.11, 16, 17.

Respecte a falsos positius, quan la il·luminació es adequada com en aquest cas no m'he trobat amb falsos positius. Però com he estat utilitzant frames independents no es te la certesa suficient del que passaria en un vídeo on hi ha una quantitat significativa de mes frames.

La Fig.18 no ha estat detectada ja que com podem apreciar la il·luminació es variada a causa d'una ombra, aquesta fa que el programa no la reconegui com un objecte blanc i la ignori completament.

Fig. 16: Senyal d'orientació detectada en un dia de sol

Fig. 17: Senyal d'indicació detectada en un dia de sol

Fig. 18: Senyal no detectada en un dia de sol

Partint d'un vídeo en el que es de nit, i per tant no hi ha una bona il·luminació aquest sistema no és eficient ja que les senyals al no estar ben il·luminades el programa no reconeix els seus colors, i per tant no les detecta.

5 Conclusions

Com hem vist als resultats obtinguts, la detecció de senyals es produeix correctament i ens fa veure que el sistema te un gran potencial, malgrat es troben algunes mancances en alguns casos. El fet de que en una carretera es puguin observar tants colors, ja siguin edificis, cotxes, altres senyals, etc, provoca que es dificulti la detecció de senyals de trànsit.

Un altre aspecte important és la il·luminació, ja que una mala il·luminació pot variar molt els colors que han de ser detectats, de manera que caldria algun sensor que detectés la il·luminació en aquell moment i s'ajustessin els rangs de color en funció d'aquesta. Una possible millora que augmentaria significativament la eficàcia en aquest aspecte, seria fer servir càmeres polaritzades les quals filtren la llum solar i els reflexes que aquesta produeix, de manera que les imatges serien més precises.

També es una realitat que s'han de tenir en compte molts aspectes, per exemple en una ciutat es poden observar moltes senyals en els carrers, però no totes estan indicant la seva funció en el carril en que et trobes, es per això que també s'hauria de tenir en compte la orientació d'aquestes senyals, és a dir, a quin carril o carrer van dirigides.

En definitiva els resultats obtinguts son adequats als que podríem esperar, però caldria tenir en compte molts aspectes per tal de ser completament efectiu. És per això que l'apartat següent és una llista de possibles línies de continuació per aquest treball de manera que s'aprofites el seu potencial completament.

6 Línies de continuació

En aquest apartat s'esmenten les possibles línies de continuació en un futur, és a dir , millores al sistema actual les quals aprofitarien el potencial d'aquest i proporcionarien una millor detecció de senyals de trànsit.Ja sigui optimitzant les funcions actuals, afegint-ne de noves,etc.

- Un augment de la resolució de les imatges per a una millor detecció, tant de les senyals com de l'interior d'aquestes. Això permetrà que el reconeixement *OCR* sigui més efectiu i per tant no apareguin falsos positius o falli a l'hora de detectar senyals.
- Seguint amb l'anterior punt, aquesta millora no formaria part directament del codi actual, sinó que aniria més enfocat a millorar la eficiència del reconeixement de caràcters , ja sigui entrenant un model propi, o millorant el *pytesseract* actual.
- Entrenar un model per al reconeixement de senyals, de manera que aquest confirmi que la senyal detectada mitjançant *HSV* és realment una senyal de trànsit o no, d'aquesta manera es podria ampliar el rang de colors per a la de detecció, i gràcies al model entrenat s'eliminarien objectes no desitjats [23].
- Aplicar sensors d'il·luminació a l'hora de gravar les imatges, de manera que en funció de la il·luminació els rangs de colors per a la detecció de senyals varii.
- Classificar les senyals d'indicació en funció del tipus d'aquesta, ja sigui una senyal de reducció de carrils, de gir a la dreta, de pas de vianants, etc, i que el programa mostri el tipus i no es limiti només a mostrar la senyal. D'aquesta manera es podria fer que un cop detectat el tipus de senyal d'indicació, en comptes de mostrar la imatge captada es mostrés una imatge en bona resolució ja definida per defecte per a cada tipus d'aquestes senyals.
- Ampliar la detecció a altres senyals com poden ser les de velocitat, obligació, perill, semàfors, etc.
- Introduir un sistema de guia el qual et vagi dient que has de fer en funció de les senyals que va detectant, o que faci que el propi vehicle reconegui que ha de fer, això últim seria útil per a cotxes amb conducció autònoma.
- Utilitzar una càmera polaritzada per a la gravació de frames, de manera que els efectes que provoquen els rajos de llum no variïn tant els colors de la imatge, i facilitin la detecció de senyals [24].

7 Agraïments

Aquests 4 mesos en els quals he treballat per al desenvolupament d'aquest projecte, certes persones m'han donat la força i el suport per seguir endavant, i es per això que trobo

just redactar aquesta secció per tal d'agrair el que han fet per mi.

Primer de tot, m'agradaria agrair als meus pares per la seva paciència i ànims que m'han estat donant durant tot el procés. M'han recolzat enormement en els bons i mals moments que un treball d'aquest calibre pot comportar. Sense ells no se si hagués tingut la força per seguir endavant.

També m'agradaria agrair a la meva parella la qual ha estat una gran font d'inspiració, ha sabut com animar-me i fer-me sentir amb ganes de treballar durant tot aquest temps.

La persona a la qual he d'agrair gran part de la feina feta es el meu tutor Jordi Serra, ja que sense la seva valuosa ajuda, no hagués sabut per on tirar. Els coneixements i consells que m'ha aportat han sigut de vital importància, i m'han brindat les eines necessàries per realitzar i completar el meu treball satisfactòriament.

Seguidament vull agrair als meus companys que han estat a prop meu, els quals m'han donat consells quan estava en moments de bloqueig, i m'han animat directa o indirectament.

Per últim i no menys important agrair a la universitat i a Jordi Pons per les facilitats que proporcionen als estudiants, les guies, rúbriques, informació, plantilles, etc. Totes aquestes coses que no es veuen al final del treball son vitals per a la seva realització i sense les quals no seria possible portar a terme el TFG.

Moltes gràcies a tots!

Referències

[1] https://www.pcmag.com/news/what-is-computer-vision

[2] https://es.motor1.com/news/338087/niveles-conduccion-coche-autonomo

[3] https://www.python.org

[4] https://opencv.org

[5] https://numpy.org

[6] https://docs.python.org/3/library/pathlib.html

[7] https://docs.python.org/3/library/os.html

[8] https://www.techbeamers.com/python-glob

[9] https://python.doctor/page-tkinter-interface-graphique-python-tutoriel

[10] https://pillow.readthedocs.io/en/stable

[11] https://www.computerhope.com/jargon/g/gui.htm

[12] https://pypi.org/project/pytesseract

[13] https://zetcode.com/python/multiprocessing

[14] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html

[15] https://docs.opencv.org/master/d5/d69/tutorial_py_non_local_means.html

[16] https://www.tutorialspoint.com/opencv/opencv_median_blur.htm

[17] https://medium.com/beyondlabsey/creating-a-simple-region-of-interest-roi-inside-a-video-streaming-using-opencv-in-python-30fd671350e0

[18] https://definicion.de/rgb

[19] https://youtu.be/3D7O_kZi8-o

[20] https://www.geeksforgeeks.org/python-thresholding-techniques-using-opencv-set-1-simple-thresholding

[21] https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

[22] https://www.tutorialkart.com/opencv/python/opencv-python-gaussian-image-smoothing

[23] https://jrodthoughts.medium.com/types-of-artificial-intelligence-learning-models-814e46eca30e

[24] https://es.wikipedia.org/wiki/Filtro_polarizador

Apèndix

En aquesta secció mostraré informació variada la qual ha ajudat a la realització del TFG, i algunes imatges de resultats mencionats anteriorment.

Les següents url son referències les quals no van senyalades en el text del projecte però han estat útils.

- Instal·lació pytesseract-OCR

  GitHub - madmaze/pytesseract: A Python wrapper for Google Tesseract

- Guia per a la realització d'una interfaç d'usuari

  https://youtu.be/YXPyB4XeYLA

- Conversor de color RGB-HSV

  https://www.rapidtables.com/convert/color/rgb-to-hsv.html

Cal recordar que el codi del projecte es pot veure en un repositori github privat, al qual si es vol tenir accés s'ha d'enviar un correu a guillem.campo@e-campus.uab.cat.

Com he mencionat a la secció de resultats, en aquesta secció es mostren els resultats obtinguts d'un vídeo en el qual el dia està plovent i amb núvols al cel. Podem apreciar que hi ha senyals que s'han trobat correctament i falsos positius els quals el programa ha confós com a senyals reals.

Fig. 19: Senyals detectades en dia de pluja

Fig. 20: Senyals detectades en dia de pluja

Fig. 21: Senyals detectades en dia de pluja

Fig. 22: Senyals detectades en dia de pluja