Detecció i identificació de números de loteria a partir de fotografies

Martí Caixal i Joaniquet, Ricard López Olivares

Abstrac

This project aims to create a program that is able to identify the lottery numbers from photographies. As of today, there is no specific software whose main purpose is this one. Nontheless, there are a bunch of multi-purpose OCR programs out there. This project uses multiple feature detection methods to first transform the picture, followed by image processing and later applying a specific OCR for this type of font created by us. The final result is quite good, yet it is unable to work properly for all kinds of pictures.

Keywords — OCR, Homography, Feature detector, image comparison, MATLAB.

1 Introducció

Per aquest projecte de Visió per Computador del grau d'Enginyeria Informàtica es vol fer un treball per poder saber els números d'un tiquet de loteria o similar a través d'una fotografia.

2 Objectius i proposta

L'objectiu principal del projecte és, com ja s'ha comentat, poder obtenir els números d'un tiquet de loteria a través d'una imatge. Si més no, es pot separar en diferents subojectius independents els uns dels altres:

- Transformar imatge: La fotografia de cada tiquet cal esperar que serà sempre agafada des d'un angle diferent. Així doncs, cada imatge tindrà una mida diferent, el tiquet estarà amb una orientació diferent, i en general no hi haurà correspondència entre les diferents imatges. L'objectiu d'aquest subapartat és aplicar una homografia per tal d'aconseguir una transofmació per cada imatge que ens permet tenir-les totes sobre un mateix pla. És a dir, que un cop s'hagi aplicat aquesta transformació, totes les imatges apareixen com si s'haguessin agafat des d'un mateix angle.
  Per aconseguir fer aquesta homografia, caldrà tenir una imatge base i buscar uns punts de referència amb la resta d'imatges. La idea és fer-ho de forma automàtica i no pas manualment seleccionant els punts iguals d'ambdues imatges. D'aquesta manera s'aconsegueix que funcioni molt més ràpid per qualsevol imatge d'entrada que sigui d'un mateix tipus de tiquet de loteria

- Processar imatge: També s'ha de fer algunes correccions a les imatges. El més probable és que, quan es faci una binarització per passar-ho a blanc i negre i ens quedem només amb la informació de la imatge que ens interessa, hi hagi imperfeccions. Per tant, caldrà corregir-les mitjançant, segurament, operacions morfològiques. Tot i això, fins que no ens trobem amb el problema, no es pot saber exactament quin tipus de correccions es necessitarà.

- Separar números: Un cop es tenen els números identificats, la idea és separar-los en imatges independents per poder tractar amb ells individualment.

- Detectar números: El subobjectiu final és el més important, poder arribar a detectar quins números tenen els ticket de la loteria. Com que ara ja es té cada dígit del tiquet per separat, la idea per complir l'objectiu és veure la correlació que té cada un d'ell amb els 10 dígits que es tenen de base. El dígit que més correlació tingui serà el que dirà la predicció.

3 Estat de l'art

Actualment no hi ha cap sistema que haguem trobat que es dediqui a fer exactament el que vol fer el nostre projecte.

Sí que hi ha, però, moltes eines i treballs ja realitzats que compleixen algunes de les funcionalitats que necessitem nosaltres.

L'exemple que més hem trobat que ja està implementat i és quelcom similar, és la detecció dels números i lletres de les matrícules dels vehicles. A la pràctica és molt semblant a la detecció dels dígits a un bitllet de loteria. La majora d'aquest projectes fan la part de visió per computador i després fan servir xarxes neurals per aconseguir saber quin número és cada dígit. [1]

També s'ha trobat que ja existeixen mètodes per trobar punts de referència entre dues fotografies. Mitjançant aquests punts és com es pot trobar la transformació necessària per DLT. [2]

3 Dades: exemples i fonts de dades

Per realitzar aquest projecte es necessita un conjunt d'imatges de mostra. Per internet hi ha moltes fotografies que ja compleixen amb les nostres necessitats: vàries angles, varies qualitats i resolucions, etc… Addicionalment, un membre del grup té una àvia que guarda els números de loteria que ha jugat. Això ens permet aconseguir fotografies més exactes o específiques per fer les proves que faci falta.

A continuació es mostren dos fotografies d'exemple:

4 Experiments, Resultats i Anàlisis

El treball, en un principi, s'intenta abordar de manera que s'assoleixi un resultat final completament automàtic. Més

endavant es veurà com en determinats casos no ha pogut ser així, quines han sigut les causes i com s'ha fet front a la qüestió.

Experiments

En aquest apartat s'explica el seguit de proves i experiments realitzats, punt a punt.

Homografies

Aquestes primeres proves consisteixen en generar homografies per tal de transformar la imatge. Per fer-ho és necessari una imatge base que tingui l'orientació, rotació i escalat desitjat.

SURF

El primer intent d'homografia es fa mitjançant el mètode SURF[3]. El resultat, però, és un problema que s'arrossega durant uns quants intents amb altres mètodes. Es tracta dels fons de les imatges quan estan carregats, és a dir, tenen detalls al fons, números, etiquetes, etc. Els punts automàtics que troba són molts i, precisament, la majoria no es repeteixen entre la imatge base i la que cal transformar.

Fig 3. Resultat primera homografia Surf

Harris

El segon mètode provat és l'anomenat Harris Corner Detector. Aquesta part s'ha afrontat de dos maneres diferents. La primera és a partir de la funció ja implementada al MATLAB, i la segona programar el nostre propi detector de Harris.

La primera opció, la utilització de Harris [4] amb la funció de MATLAB, anomenada detectHarrisFeatures(). Funciona de manera que se li passa una imatge i et retorna totes les cantonades de la imatge proporcionada. Per tant, és necessari fer la crida per cada una de les imatges i, seguidament, fer un "match" de punts. En aquest punt tenim una funció anomenada "matchFeatures()" de MATLAB, la qual busca els punts que més s'assemblin entre ells. Tal com s'ha mencionat abans, el problema persisteix i s'obté el següent resultat:

a) Visualitzacio del punts que fan match

b) Resultat de l'homografia

Fig 4. Homografia automatica Harris.

Veient el resultat anterior, es decideix programar el detector de Harris nosaltres mateixos. La principal raó de fer-ho així és degut al "match", aquest es queda amb molt pocs punts, tal com es pot veure en la figura anterior.

Per implementar Harris es necessita les derivades X i Y de la imatge, podent-se aconseguir de diverses maneres. La primera és fer una convolució amb un filtre, aquest pot ser gaussià, box , etc. La segona és fer una fórmula matemàtica (transformada de Fourier o DFT), aplicant un filtre com si fos una escala, tant per l'eix vertical ([1,0,-1;1,0,-1;1,0,-1]), com per l'horitzontal ([1,1,1;0 0 0;-1 -1 -1]).

Un cop es té les derivades de la imatge, es multipliquen entre elles i seguidament s'eleva el resultat al quadrat. D'aquesta manera s'obté el tensor superficial:

$$A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$

Per acabar, amb l'ajuda del tensor, es pot obtenir les cantonades i dir si un punt canvia molt. En cas afirmatiu, vol dir que és una cantonada. Cada punt té un resultat anomenat R indicant si canvia molt. Tenint un valor per cada píxel de la imatge, s'aplica un "threshold" per quedar-se amb els punts més significatius a aquest canvi.

El resultat d'aquest nou detector de Harris és el següent:

Fig 5. Resultat de Harris

Això es realitza amb les dues imatges i, a continuació, es busca els punts que fan "match" entre ells. Aquesta relació es fa comparant els descriptors de cada punt. Per aquest cas, són una finestra de 3x3 on el centre és el punt significatiu de Harris. Cada descriptor es compara amb tots els de l'alta imatge, sent el que doni menys diferència el que fa "match".

Finalment, es fa l'homografia, però, com es pot veure a la següent figura, el resultat continua sent dolent.

Fig 6. Resultat de la homografia Harris

Homografia a mà

Veient que el problema és molt difícil de solucionar en segons quines imatges, es decideix fer una homografia mitjançant 4 punts per imatge escollits manualment. Tot i ser-ne 8 en total, els 4 de la imatge base són sempre els mateixos i no cal seleccionar-los a cada execució, estan ja guardats al disc. És important que els punts es seleccionin sempre en el mateix ordre.

Tal com podem veure en la figura següent aquesta homografia és la més pertinent, ja que es mostra com olem i el tema de l'afinitat en una imatge s'elimina.

Fig 7. Resultat de l'homografia escollint 4 punts

També cal mencionar que aquesta eliminació de l'afinitat és bastant consistent i funciona en totes les proves fetes amb diferents imatges.

Així doncs, al final s'opta per fer dos homografies, la primera escollint els punts i la segona mitjançant SURF amb els edges (arestes / bores) de cada imatge.

Fig 8. Exemple de Edge

La següent figura mostra el resultat final. El tiquet de l'esquerra és el base, mentre que el de la dreta és el tiquet transformat.

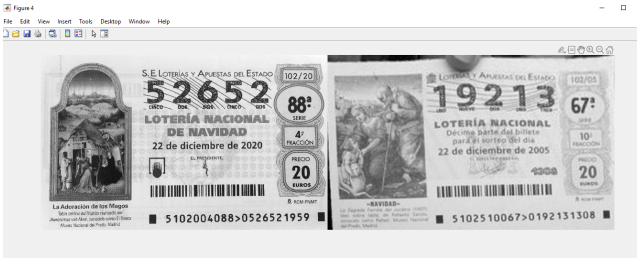

Fig 9. Resultat final

Retall de la imatge

Tenint la imatge ben transformada, el pròxim pas és fer una aproximació d'on hi ha la informació necessària per poder assolir l'objectiu. La mateixa naturalesa del disseny dels tiquets permet fer una molt bona aproximació per tots els casos. La zona que conté els números sempre es troba a la mateixa localització i, gràcies a la transformació prèvia, es pot directament retallar la imatge.

Aleshores, amb l'ús de la funció "imcrop", es pot simplement indicar els 2 punts i 2 distàncies que delimiten la zona amb el codi numèric

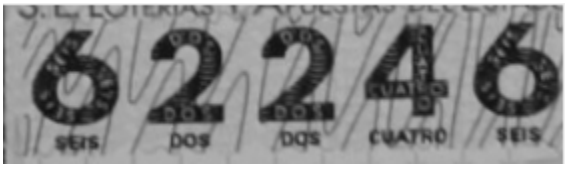

Fig 9. Imatge retallada

Aquest fet facilita molt, com ja es veurà, les tasques que hi ha a continuació

Binarització i operacions morfològiques

La binarització de la imatge consisteix en discretitzar tots els píxels en 2 categories, blancs i negres, de manera que, tot i passar a tenir dos colors, es pugui diferenciar entre totes les parts principals de la imatge.

El mètode estàndard es fa a través d'un punt de tall arbitrari. Si prèviament s'ha vist la imatge, és fàcil trobar-lo. El problema apareix quan cada imatge té una il·luminació diferent i, per tant, el punt de tall canvia. Aquí entre en joc la utilització del mètode Otsu[6], el qual tria un "threshold" que minimitza la variància intra-classe entre els píxels blancs i els negres. Per aconseguir fer-ho s'utilitza la funció "graythresh" de MATLAB. D'aquesta manera es pot trobar automàticament el valor òptim per fer la binarització.

La següent figura mostra com queda la imatge un cop fet-a la binarització.

Fig 10. Imatge binaritzada

Tot i això, hi ha bastant soroll (paraules extra, punts blancs sobre el fons negre, etc…). Mitjançant operacions morfològiques[7] s'aconsegueix millorar el resultat. Primer es fa una erosió per tal d'eliminar els petits punts que hi ha al fons negre i separar àrees que hagin pogut quedar enganxades. Seguidament, al resultat obtingut s'aplica un "close" per omplir tots els forats que hi ha dins de les regions i fer-les completament sòlides.

La següent figura mostra el resultat un cop corregides les imperfeccions. Si bé és veritat que segueix havent-hi soroll, ara els números ja estan tots ben definits i separats de les altres regions. Més endavant es veu com diferenciar les regions

importants de les que no ho són.

Fig 11. Correcció correcte

Malauradament, hi ha algunes situacions on és impossible arreglar les imperfeccions de la imatge. Les que tenen ombres o estan mullades són un exemple. Un humà pot utilitzar el raonament i esbrinar els valors tot i veure només part del número. Però un ordinador no ho pot fer de forma tant senzilla i, per tant, aquests casos no es pretenen resoldre.

Fig 12. Correcció incorrecte

La següent figura mostra un altre resultat. Es pot veure que s'aconsegueix un resultat robust per la gran majoria de tiquets, sempre que aquests tinguin una fotografia original mínimament acceptable.

Fig 13. Correcció correcte

Localització de dígits

Un cop es té la imatge postprocessada correctament, cal localitzar els dígits individualment.

La idea general per localitzar-los és mitjançant l'eina "regionprops"[8] de Matlab. S'encarrega de buscar totes les regions blanques contigües dins de la imatge donada. Tot i això, el resultat inicial és molt dolent, ja que simplement busca totes les regions existents.

Fig 14. Regions mal localitzades

Així doncs, la primera millora realitzada és filtrar cada una d'aquestes regions per la seva àrea. Pels dígits mostrats fins ara funciona bé. Provant amb altres tiquets es veu que únicament filtrant per l'àrea no es pot solucionar el problema per tots els casos.

Les imatges no sempre tenen la mateixa il·luminació i, per tant, les operacions morfològiques aplicades no sempre aconsegueixen el millor resultat. A continuació es pot veure una prova realitzada que no aconsegueix identificar correctament les regions. La següent imatge mostra un resultat on hi ha regions identificades amb àrees més grans que alguns dígits. Per tant, resulta impossible trobar un "threshold" per identificar dígits mitjançant l'àrea.

Fig 14. Regions mal localitzades

La següent millora es basa en la posició i forma de les regions. Sabent que sempre són dígits i tenen la mateixa font, es pot filtrar per forma de manera que l'amplada sempre sigui més petita que l'alçada. Addicionalment, també es té en compte que les regions aconseguides estiguin totes sobre una mateixa alçada respecte a la imatge sencera.

La següent figura mostra el resultat un cop havent aplicat les millores esmentades.

Fig 15. Regions ben localitzades

Reconeixement de dígits

Tot i tenir identificades les parts de la imatge que componen cada dígit, encara és necessari reconèixer el significat de cada un d'ells.

La primera manera en la qual es du a terme la tasca és mitjançant el paquet "ocr"[9] implementat al mateix Matlab. Entrant cada regió com a input a la llibreria, no obté cap resultat. En canvi, si s'afinen els paràmetres de configuració (characterSet = 0123456789, TextLayout=Character), és capaç de trobar el valor correcte en la majoria dels casos. El problema està quan no s'ha pogut assolir un bon resultat durant les operacions morfològiques. L'última figura, tot i que a simple vista veure's perfectament cada dígit, es pot observar que els dígits 1, 9 i 2 tenen el peu unit a petites taques que realment són soroll. La il·luminació a l'hora de fer fotografies moltes vegades no es pot controlar, i, per tant, aquest soroll és molt difícil treure per tots els casos.

Veient que la llibreria "ocr" no és idònia, es decideix programar un sistema de comparació entre dígits. Ara es juga amb l'avantatge que sempre s'utilitza la mateixa font, les imatges ja estan amb una bona orientació i només hi ha 10 classes. Així doncs, s'extreu un dígit de cada classe que serveix de base per la resta.

Fig 16. Imatges base per comparar

La comparació és relativament senzilla. Cada dígit localitzat és comparat amb els 10 dígits base. El primer pas és redimensionar el dígit localitzat de manera que tingui la mateixa mida que el dígit base amb el qual es compara. Tot i pensar que això pot comportar que el dígit localitzat es vegi afectat per un canvi de proporcions, realment s'està davant d'avantatge. Els dígits que no siguin els que pertoquen, es veuran afectats per la redimensió, però el dígit que realment és el correcte es veu afectat de forma insignificant.

Seguidament es procedeix a fer la comparació entre les dos imatges. S'ha realitzat proves amb "Mean Squared Error", "Peak Signal to Noise Ratio" i "Structural Similarity"[10].

Utilitzant aquest nou mètode el resultat final sí és satisfactori i permet identificar correctament el valor de cada dígit.

Resultats

La taula següent mostra el percentatge de dígits que s'identifiquen correctament usant els diferents mètodes explicats. Es veu com els creats manualment, a l'estar fets específicament per aquest cas i comptar amb dades d'entrenament molt semblants a les de test, aconsegueix uns resultats molt bons. Els únics casos en els quals falla són els vistos anteriorment on ombres o reflexes fan impossible la detecció dels dígits.

OCR	MSE	PSNR	SSIM
73%	98%	98%	97%

Taula 1. Resultats

5 Conclusions

Com a conclusió extraiem que no totes les imatges són idíl·liques, les millors són les que estan més preparades com les que no tenen fons o un fons uniforme. Aquestes les considerem com massa idíl·liques que serveixen per fer testos i proves bàsiques, però per veure si funciona realment o mirar els límits que es poden arribar són millor les imatges que pugui ser un repte com les imatges amb un fons molt carregat.

També es pot extreure que tot que podem trobar implementat, com en una funció, o algun toolbox, etc. no sempre és el millor, a vegades és millor implementar-ho per tu mateix i trobaràs millors resultats com ens ha passat amb la detecció de números.

Per acabar, també cal comentar que és millor que un conjunt de mètodes diferents que un únic mètode diverses vegades.

Bibliografia

[1] Sergio Theodoridis, Konstantinos Koutroumbas, "Patther Recognitions" 3 edition.

[2] Sandra Kief, Danie Nuen, the Power of the Weisfeiler – Leman Algorithm to Decompose Graphs, SIAM Journal on Discrete Mathematics, 10.1137/20M1314987, 36, 1, (2022).

[3] Bay, H., A. Ess, T. Tuytelaars, and L. Van Gool. "SURF:Speeded Up Robust Features." Computer Vision and Image Understanding (CVIU).Vol. 110, No. 3, pp. 346–359, 2008.

[4] Harris, C., and M. Stephens, "A Combined Corner and Edge Detector," Proceedings of the 4th Alvey Vision Conference, August 1988, pp. 147-151.

[5] Pattern Recognition Letters Volume 32, Issue 14, 15 October 2011, Pages 1714-1719

[6] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms." *IEEE Transactions on Systems, Man, and Cybernetics*. Vol. 9, No. 1, 1979, pp. 62–66.

[7] Gonzalez, Rafael C., Richard E. Woods, and Steven L. Eddins. *Digital Image Processing Using MATLAB*. Third edition. Knoxville: Gatesmark Publishing, 2020.

[8] Shoemake, Ken, *Graphics Gems IV*. Edited by Paul S. Heckbert, Morgan Kaufmann, 1994, pp. 222–229.

[9] Smith, R., D. Antonova, and D. Lee. Adapting the Tesseract Open Source OCR Engine for Multilingual OCR. Proceedings of the International Workshop on Multilingual OCR, (2009).

[10] Zhou, W., A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. "Image Quality Assessment: From Error Visibility to Structural Similarity." IEEE Transactions on Image Processing. Vol. 13, Issue 4, April 2004, pp. 600–612.