Universitat Politècnica de Catalunya

Visió per Computador

Short project: Reconeixement de mirades

Mario Fernández Villalba

Jordi Fructos Hernández

Group 11

Q1 2017-2018

Contents

1 Sinopsis del projecte

L'objectiu d'aquest projecte es reconèixer la mirada en una foto. El programa que hem creat marcarà els ulls de una foto donada, si existeixen, i ens dirà si estan mirant a la càmera o no.

Per fer això fa servir tècniques de Visió per Computador i Aprenentatge Automàtic. Concretament, la seva implementació conté dos classificadors: un que reconeix ulls i altre que reconeix mirades.

2 Introducció

Durant els últims anys la Visió per Computador ha anat guanyant progressivament importància en la nostra vida gracies als avenços en la capacitat computacional dels processadors i el descobriment de nous algorismes. Una de les seves aplicacions és el reconeixement facial gràcies al famós algoritme Viola-Jones [1].

En aquesta projecte volem aplicar una idea similar a la de Viola-Jones: en comptes de reconèixer cares en una foto, reconèixer els ulls i indicar-ne si estan mirant a càmera o no. En altres paraules, utilitzar la Visió per Computador per reconèixer mirades.

3 Funcionament del reconeixedor de mirades

El nostre reconeixedor de mirades agafa una imatge com a entrada, i com a sortida marca els ulls, si n'hi ha, i indicarà si estan mirant a càmera. Per fer es divideix el procés en diferents fases.

Primerament s'ha de comprovar l'existència d'ulls en l'imatge. Per fer això es parteix la imatge de entrada en subimatges de mida (32 x 32) amb una freqüència de sampling de 4 píxels i es guarda la posició de cada subimatge en l'imatge original. A continuació cada subimatge obtinguda es re-escalada a les dimensions (64 x 64), ja que el classificador ha sigut entrenat amb imatges d'aquesta mida, i se'n extreuen les característiques. Finalment es classifiquen les imatges en ulls i no ulls. A la Figura 1 podem veure un exemple d'imatge de entrada del reconeixedor, i a la Figura 2 marcades en groc les posicions dels possibles ulls en aquesta.

Figura 1: Exemple d'entrada del reconeixedor de mirades.

Figura 2: Possibles ulls marcats a l'entrada del reconeixedor.

El següent pas es extreure la posició exacta dels ulls dins de la imatge. Com hem pogut observar, es detecten una gran quantitat de subimatges com a possibles ulls, i es del nostre interes extreure'n només dos. Per fer això hem considerat dos enfocaments diferents:

1. El primer enfocament ha sigut el d'expandir tots el possibles ulls utilitzant una màscara. Mitjançat aquesta màscara, que consisteix en un "disc" amb major pes al centre i menor als extrems, hem aconseguit diverses zones connexes de les que hem calculat la posició dels centroides. Finalment, hem comparat aquest entre ells per buscar una parella que estigui alineada horitzontalment, posats que en general els ulls estan alineats. Cal dir, que si no es detecta cap parella alineada o només hi ha un possibles ulls per comparar, no podrem afirmar que l'imatge conté algun ull. A la Figura 3 podem observar les posicions del ulls de la imatge de la Figura 2 marcades en verd després d'aplicar aquesta idea.

Figura 3: Ulls marcats a l'entrada del reconeixedor utilitzant el primer enfocament.

2. L'altre enfocament ha sigut el d'aplicar clustering. Mitjançat el clustering jeràrquic hem dividit les posicions dels possibles ulls en clústers, escollint com a criteri per tallar el dendrograma una distància de 5. Després d'això hem calculat la posició dels centroides dels clústers mitjançant l'algoritme *k-means*. Finalment, hem calculat les rectes per a cada parell de centroides i ens hem quedat amb els centroides que tenen la recta amb menor angle. La rao d'això es que en general els ulls estan alineats perfectament horitzontalment, per tant la recta que els uneix te una inclinació ≈ 0. Cal notar que en cas de que no hi hagi cap recta o que l'angle mínim entre rectes sigui $\geq 20^{o}$, donem per suposat que no existeixen ulls a l'imatge. A la Figura 4 podem veure els centroides dels clústers corresponents a l'imatge de la Figura 2, i a la Figura 5 podem veure els centroides seleccionats com a posició dels ulls marcats en verd.

Figura 4: Centroides marcats a l'entrada del reconeixedor.

Figura 5: Ulls marcats a l'entrada del reconeixedor utilitzant el segon enfocament.

Com es pot observar els dos enfocaments donen resultats molt similars, aixi que es indiferent quin dels dos utilitzem.

Un cop trobades les posicions reals dels dos ulls es passa a classificar les subimatges d'aquests en ulls que miren a càmera i ulls que no miren a càmera mitjançant el classificador de mirades. Si la classificació dels dos ulls es la mateixa aleshores es dona aquesta com la bona per la imatge; si hi ha empats la classificació general serà de que la persona no mira a càmera però igualment s'especificarà quin es l'ull que mira a càmera. A la Figura 6 podem veure marcats en blau els ulls de l'imatge de la Figura 1 que miren a càmera.

Figura 6: Entrada del reconeixedor classificada com mirant a càmera.

4 Classificació d'ulls

4.1 Procés d'aprenentatge

Com hem comentat en la Secció 1 primerament hem de poder dir si donada una imatge aquesta representa un ull o no. Per fer això hem escollit un dels mètodes mes ràpids de Bagging, el TreeBagger. A continuació passarem a explicar el procés d'aprenentatge; explicant les dades utilitzades en aquest i les característiques escollides.

4.1.1 Dades utilitzades en l'aprenentatge

En Aprenentatge Automàtic es requereix de un gran nombre de dades per tal de entrenar un bon classificador i posteriorment testejar-lo correctament. En el nostre projecte hem obtingut aquestes dades, que en el nostre cas són imatges, de la URL https://www.bioid.com/About/BioID-Face-Database. Aquest dataset conté 1521 imatges en nivells de gris de 384 286 pixel extretes de diferents conferencies de vídeo. Aixi mateix, també conte la posició exacta dels ulls que apareixien en cada imatge.

Utilitzant les posicions dels ulls hem extret subimatges del dataset que contenen aquests, degut a que en aquesta fase del projecte només ens hem centrat en dir si una imatge representa un ull o no. Això ha generat un total de 3042 imatges d'ulls. Així mateix, també requerim imatges que no contenguin ulls per tal de donar al classificador casos negatius. Per extreure les imatges sense ulls hem agafat 18 subimagtes de cada imatge del dataset original assegurant-nos de que dintre d'aquestes no hi hagués cap píxel corresponent a un ull (per fer això hem utilitzat de nou les coordenades dels ulls). D'aquesta manera hem aconseguit mantenir una relació de 10% d'imatges amb ulls i 90% d'imatges amb no ulls.

4.1.2 Característiques utilitzades en l'aprenentatge

Un cop obtingudes les imatges requerides en l'aprenentatge, es requereix seleccionar les característiques més rellevants que defineixen als ulls. En el nostre cas hem decidit que les millors son les característiques relacionades amb els nivells de grisos i els histogrames de la orientació dels gradients, també coneguts com característiques de HOG [2].

Per obtenir les característiques relaciones amb els nivells de grisos en una imatge primerament hem calculat el nivell de gris mitjà en la imatge. A continuació hem sumat per files i per columnes els nivells de grisos i hem restat el nivell mitjà calculat anteriorment a cada suma. D'aquesta manera, d'una imatge de mida $i\,x\,j$ s'extreuen $i + j$ característiques (una per cada fila i columna). Hem considerat que el nivell de grisos pot ser rellevant degut a que per la composició de l'ull la part de la pupil·la i l'iris es molt més fosca que l'escleròtica, cosa que genera molt de contrast. A més el major contrast es dona al centre de l'ull, per això hem trobat interessant extreure les característiques per files i per columnes.

Per obtenir les característiques de HOG hem utilitzat la funció de Matlab *extractHOGFeatures*. Aquestes característiques son simplement histogrames que indiquen les orientacions dels gradients de una imatge. Concretament, es divideix la imatge en cel·les i se n'extreuen les característiques d'aquestes. Com hem comentat abans l'ull te molt de contrast a la part central, per tant es natural pensar que les direccions dels gradients es distribueixen de manera uniforme en tota la circumferència ja que el ull es circular.

Un cop definides les característiques, passem a veure l'ajust dels paràmetres.

4.2 Estudi paramètric del classificador

Degut a que utilitzem TreeBagger i que utilitzem les característiques de HOG per classificar cal ajustar una serie de paràmetres per tal d'aconseguir els millors resultats en el nostre classificador. Concretament, aquests paràmetres son el nombre d'arbres utilitzats en el TreeBagger, la mida de cel·la de HOG, la mida de bloc de HOG, i el nombre de contenidors de HOG.

Aleshores per trobar els valors "òptims" hem dissenyat dos grups d'experiments per paràmetre, que utilitzen un total de tres subsets d'imatges. Aquests subsets s'han generat tal i com s'ha descrit a la secció anterior; un de ells conté els ulls, mentre que la resta contenen fotos sense ulls. En ambdós grups d'experiments s'agafa el subset d'ulls i un dels subsets de no ulls i es parteixen sempre de la mateixa manera: n'agafem el 70% d'aquests per training i el 30% restant pel testing. Fem dos grups d'experiments diferents per paràmetre per tal d'atenuar el factor random que hi ha al generar el subset d'imatges amb no ulls.

Passem aleshores a veure els tests específics per cada paràmetre.

4.2.1 Estudi de la mida de cel·la de HOG

Primerament vam estudiar el valor optim de la mida de cel·la per HOG. Per aixo, hem dissenyat els grups d'experiments de la següent forma:

1. En primer lloc hem acordat de testejar els valors de la mida de cel·la [8x8, 12x12, 14x14, 16x16, 20x20, 24x24].
2. Seguidament hem entrenat i predit els corresponents subsets 5 cops variant el valor de la mida de cel·la mentre que manteníem constants la mida de bloc, el nombre de contenidors i el nombre d'arbres. Es varen executar 5 cops degut a que la classificació te un factor random (TreeBagger inicialitza els primers arbres de manera random) i es requeria fer una mitjana aritmètica.
3. Un cop executat els dos grups d'experiments es va fer de nou la mitjana entre els resultats d'aquests, essent aquestes mitjanes els resultats finals dels experiments.

Els resultats obtinguts es poden observar a la Figura 7. Observem que el comportament del classificador a mesura que anava incrementant la mida de cel·la ha sigut de millorar les prediccions fins a arribar a un màxim, que es troba a la mida de cel·la 16x16. Per valors de la mida de cel·la més grans que 16x16 el classificador empitjorava dràsticament. La raó d'això es que al augmentar la mida de cel·la es poden perdre detalls baixa escala [3]. Podem concloure aleshores que la millor la mida de cel·la és 16x16.

Cell	BlockSize	NumBins	Accuracy	Eyes	No Eyes
24x24	4x4	9	98,67%	10,63%	0,323%
20x20			98,65%	10,94%	0,308%
16x16			99,74%	2,41%	0,026%
12x12			99,57%	3,94%	0,047%
8x8			99,59%	4,11%	0,010%

Figura 7: Resultats de l'estudi de la mida de cel·la de HOG.

4.2.2 Estudi de la mida de bloc de HOG

Seguidament vam estudiar el valor optim de la mida de bloc de HOG, també conegut com a nombre de cel·les de HOG. Per aixo, hem dissenyat els grups d'experiments de la següent forma:

1. En primer lloc hem acordat de testejar els valors de la mida de bloc [2x2, 3x3, 4x4, 5x5, 6x6].
2. Seguidament hem entrenat i predit els corresponents subsets 5 cops variant el valor de la mida de bloc mentre que manteníem constants la mida de cel·la, el nombre de contenidors i el nombre d'arbres. Es varen executar 5 cops degut a que la classificació te un factor random.
3. Un cop executat els dos grups d'experiments es va fer de nou la mitjana entre els resultats d'aquests, essent aquestes mitjanes els resultats finals dels experiments.

Els resultats obtinguts es poden observar a la Figura 8. Observem que el comportament del classificador a mesura que anava incrementant la mida de bloc ha sigut de millorar les prediccions fins a arribar a un màxim, que es troba a la mida de bloc 4x4. Per valors de la mida de bloc més grans que 4x4 el classificador empitjorava dràsticament. La raó d'això es que al augmentar la mida de bloc, o en altres paraules, el nombre de cel·les; es perd la habilitat de suprimir canvis de il·luminació a escala local [4]. Podem concloure aleshores que el millor nombre de cel·les és 4x4.

Cell	BlockSize	NumBins	Accuracy	Eyes	No Eyes
16x16	6x6	9	98,69%	10,61%	0,301%
	5x5		98,67%	10,76%	0,302%
	4x4		99,75%	2,30%	0,026%
	3x3		99,69%	2,91%	0,032%
	2x2		99,67%	3,05%	0,033%

Figura 8: Resultats de l'estudi de la mida de bloc de HOG.

4.2.3 Estudi del nombre de contenidors de HOG

A continuació vam estudiar el valor optim del nombre de contenidors de HOG. Per aixo, hem dissenyat els grups d'experiments de la següent forma:

1. En primer lloc hem acordat de testejar els valors del nombre de contenidors [9, 10, 11, 12, 13].
2. Seguidament hem entrenat i predit els corresponents subsets 5 cops variant el valor del nombre de contenidors mentre que manteníem constants la mida de cel·la, la mida de bloc i el nombre d'arbres. Es varen executar 5 cops degut a que la classificació te un factor random.
3. Un cop executat els dos grups d'experiments es va fer de nou la mitjana entre els resultats d'aquests, essent aquestes mitjanes els resultats finals dels experiments.

Els resultats obtinguts es poden observar a la Figura 9. Observem que el comportament del classificador és aproximadament el mateix per a diferents valors del nombre de contenidors. L'única diferència és que a més nombre de contenidors més característiques s'extrauran, per lo que s'augmentarà el cost computacional. Per això mateix, agafem com millor valor del nombre de contenidors 9.

Cell	BlockSize	NumBins	Accuracy	Eyes	No Eyes
16x16	4x4	13	99,73%	2,44%	0,029%
		12	99,72%	2,55%	0,032%
		11	99,75%	2,26%	0,035%
		10	99,75%	2,31%	0,028%
		9	99,72%	2,56%	0,028%

Figura 9: Resultats de l'estudi del nombre de contenidors de HOG.

4.2.4 Estudi del nombre d'arbres en el classificador

Finalment vam estudiar el valor optim del nombre d'arbres. Per aixo, hem dissenyat els grups d'experiments de la següent forma:

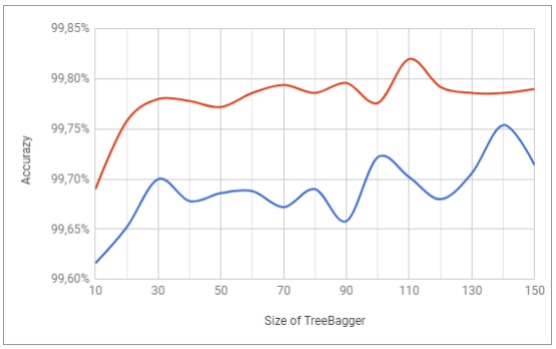
1. En primer lloc hem acordat de testejar els valors del nombre d'arbres [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150].
2. Seguidament hem entrenat i predit els corresponents subsets 5 cops variant el valor del nombre d'arbres mentre que manteníem constants la mida de cel·la, la mida de bloc i el nombre de contenidors. Es varen executar 5 cops degut a que la classificació te un factor random.
3. Un cop executat els dos grups d'experiments es va fer de nou la mitjana entre els resultats d'aquests, essent aquestes mitjanes els resultats finals dels experiments.

Els resultats obtinguts es poden observar a la Figura 10. Observem que el comportament del classificador és aproximadament el mateix per a diferents valors del nombre d'arbres. Tenim que quant més augmenta el nombre d'arbres més augmenta el temps computacional, per tant ens interessa agafar un valor que equilibri temps computacional i error obtingut. Hem agafat com millor valor del nombre d'arbres 90, ja que encara que el classificador amb aquest nombre d'arbres no dona el millor error, és més ràpid.

Cal notar que a la Figura 10b s'observen dues línies que corresponen als resultats dels dos grups d'experiments sense fer la mitjana. Ens ha semblat interessant comentar que amb el segon subset d'imatges de no ulls s'aconsegueixen millors prediccions; per tant en el nostre programa final el classificador estarà entrenat amb aquest subset.

Size	Accuracy	Eyes	No Eyes
10	99,69%	2,53%	0,071%
20	99,76%	2,21%	0,027%
30	99,78%	2,28%	0,019%
40	99,78%	2,12%	0,015%
50	99,77%	2,23%	0,010%
60	99,79%	1,99%	0,019%
70	99,79%	1,94%	0,015%
80	99,79%	2,19%	0,010%
90	99,80%	1,94%	0,012%
100	99,78%	2,08%	0,019%
110	99,82%	1,70%	0,015%
120	99,79%	1,99%	0,017%
130	99,79%	2,03%	0,015%
140	99,79%	2,01%	0,017%
150	99,79%	2,01%	0,012%

(a) Taula amb els resultats de l'estudi de la mida de l'arbre.



(b) Gràfica amb els resultats de l'estudi de la mida de l'arbre.

Figura 10: Resultats de l'estudi de la mida de l'arbre.

5 Classificació de mirades

5.1 Procés d'aprenentatge

Un cop hem obtingut les imatges corresponents als ulls cal classificar-los en si estan mirant a càmera o no. Per fer això hem creat un nou classificador de mirades mitjançant TreeBagger. A continuació passarem a explicar el procés d'aprenentatge; explicant les dades utilitzades en aquest i les característiques escollides.

5.1.1 Dades utilitzades en l'aprenentatge

Per aquest classificador simplement hem aprofitat les imatges d'ulls obtinguts en la classificació d'ulls. Hem creat un petit programa amb el que assignar dues classes a les imatges existents, ull que mira a càmera o ull que no mira a càmera.

5.1.2 Característiques utilitzades en l'aprenentatge

Un cop assignades les classes, hem passat a escollir les característiques més rellevants que defineixen a un ull que mira a càmera. En el nostre cas, al igual que en classificador d'ulls hem decidir de nou que les millors son les característiques relacionades amb els nivells de grisos i els histogrames de la orientació dels gradients.

Per obtenir les característiques relaciones amb els nivells de grisos hem operat de manera similar a la que s'ha explicat en la Secció 4.1.2. Hem considerat que el nivell de grisos pot ser rellevant degut a que quan un ull mira a càmera la pupil·la i el iris queden centrats en l'imatge, mentre que quan no mira queden descentrats. En altres paraules, la concentració de grisos a l'imatge quedara descentrada si el ull no mira a càmera. Noteu com a la Figura 11a la major concentració de grisos es al centre, i com a la Figura 11b es a la dreta.

(a) Ull que mira a càmera.

(b) Ull que no mira a càmera.

Figura 11: Exemples de fotos d'ulls.

Per obtenir les característiques de HOG hem operat de manera similar a la que s'ha explicat en la Secció 4.1.2. Tenim que quan un ull mira a càmera el conjunt pupil·la-iris es molt més circular que quan no mira. A la Figura 11 es pot visualitzar aquesta idea. Aleshores, hem considerat que les característiques de HOG poden ser rellevants perquè en certa manera "mesuren" la circularitat representada en una imatge amb l'orientació dels gradients.

Destaquem que per mesurar la circularitat pot ser natural pensar en utilitzar la transformada de Hough [5] per a detecció de cercles. Vam estar barrejant aquesta idea, però després de fer un preestudi ràpid de la accuracy obtinguda amb la transformada la vam descartar.

Un cop definides les característiques, passem a veure l'ajust dels paràmetres.

5.2 Estudi paramètric del classificador

Degut a que utilitzem TreeBagger i que utilitzem les característiques de HOG per classificar cal ajustar una serie de paràmetres per tal d'aconseguir els millors resultats en el nostre classificador. Concretament, aquests paràmetres son el nombre d'arbres utilitzats en el TreeBagger, la mida de cel·la de HOG, la mida de bloc de HOG, i el nombre de contenidors de HOG.

Aleshores per trobar els valors "òptims" hem dissenyat un grup d'experiments per paràmetre, que utilitzen un total de dos subsets d'imatges. Aquests subsets s'han generat a partir del dataset descrit a la Secció 5.1.1, agafant el 70% de les imatges per a training i el 30% per a testing.

Passem aleshores a veure els tests específics per cada paràmetre.

5.2.1 Estudi de la mida de cel·la de HOG

Primerament vam estudiar el valor optim de la mida de cel·la per HOG. Per aixo, hem dissenyat el grups d'experiments de la següent forma:

1. En primer lloc hem acordat de testejar els valors de la mida de cel·la [8x8, 12x12, 14x14, 16x16, 20x20, 24x24].
2. Seguidament hem entrenat i predit els corresponents subsets 5 cops variant el valor de la mida de cel·la mentre que manteníem constants la mida de bloc, el nombre de contenidors i el nombre d'arbres. Es varen executar 5 cops degut a que la classificació te un factor random (TreeBagger inicialitza els primers arbres de manera random) i es requeria fer una mitjana aritmètica.

Els resultats obtinguts es poden observar a la Figura 12. Observem que el comportament del classificador a mesura que anava incrementant la mida de cel·la ha sigut de millorar les prediccions fins a arribar a un màxim, que es troba a la mida de cel·la 16x16. Per valors de la mida de cel·la més grans que 16x16 el classificador empitjorava dràsticament. La raó d'això es que al augmentar la mida de cel·la es poden perdre detalls baixa escala [3]. Podem concloure aleshores que la millor la mida de cel·la és 16x16.

Cell	BlockSize	NumBins	Accuracy	Looking	Not Looking
24x24	4x4	9	82.65%	11.40%	26.39%
20x20			81.88%	11.98%	27.44%
16x16			85.69%	8.53%	23.09%
12x12			84.49%	9.69%	24.35%
8x8			85.23%	9.51%	22.75%

Figura 12: Resultats de l'estudi de la mida de cel·la de HOG.

5.2.2 Estudi de la mida de bloc de HOG

Seguidament vam estudiar el valor optim de la mida de bloc de HOG, també conegut com a nombre de cel·les de HOG. Per aixo, hem dissenyat el grup d'experiments de la següent forma:

1. En primer lloc hem acordat de testejar els valors de la mida de bloc [2x2, 3x3, 4x4, 5x5, 6x6].
2. Seguidament hem entrenat i predit els corresponents subsets 5 cops variant el valor de la mida de bloc mentre que mantenïem constants la mida de cel·la, el nombre de contenidors i el nombre d'arbres. Es varen executar 5 cops degut a que la classificació te un factor random.

Els resultats obtinguts es poden observar a la Figura 13. Observem que el comportament del classificador a mesura que anava incrementant la mida de bloc ha sigut de millorar les prediccions fins a arribar a un màxim, que es troba a la mida de bloc 3x3. Per valors de la mida de bloc més grans que 3x3 el classificador empitjorava dràsticament. La raó d'això es que al augmentar la mida de bloc, o en altres paraules, el nombre de cel·les; es perd la habilitat de suprimir canvis de il·luminació a escala local [4]. Podem concloure aleshores que el millor nombre de cel·les és 3x3.

Cell	BlockSize	NumBins	Accuracy	Looking	Not Looking
16x16	6x6	9	82.25%	11.62%	27.05%
	5x5		81.84%	11.87%	27.71%
	4x4		85.78%	8.38%	23.09%
	3x3		85.80%	8.93%	22.20%
	2x2		85.38%	9.04%	23.09%

Figura 13: Resultats de l'estudi de la mida de bloc de HOG.

5.2.3 Estudi del nombre de contenidors de HOG

A continuació vam estudiar el valor optim del nombre de contenidors de HOG. Per aixo, hem dissenyat el grups d'experiment de la següent forma:

1. En primer lloc hem acordat de testejar els valors del nombre de contenidors [9, 10, 11, 12, 13].
2. Seguidament hem entrenat i predit els corresponents subsets 5 cops variant el valor del nombre de contenidors mentre que manteníem constants la mida de cel·la, la mida de bloc i el nombre d'arbres. Es varen executar 5 cops degut a que la classificació te un factor random.

Els resultats obtinguts es poden observar a la Figura 14. Observem que el comportament del classificador és aproximadament el mateix per a diferents valors del nombre de contenidors. L'única diferència és que a més nombre de contenidors més característiques s'extrauran, per lo que s'augmentarà el cost computacional. Per això mateix, agafem com millor valor del nombre de contenidors 10.

Cell	BlockSize	NumBins	Accuracy	Looking	Not Looking
16x16	4x4	13	85.65%	8.20%	23.69%
		12	85.10%	8.86%	24.19%
		11	85.49%	8.64%	23.42%
		10	85.89%	8.31%	22.92%
		9	85.76%	9.04%	22.15%

Figura 14: Resultats de l'estudi del nombre de contenidors de HOG.

5.2.4 Estudi del nombre d'arbres en el classificador

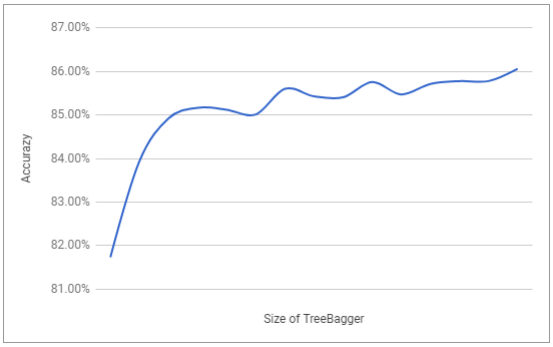
Finalment vam estudiar el valor optim del nombre d'arbres. Per aixo, hem dissenyat el grup d'experiments de la següent forma:

1. En primer lloc hem acordat de testejar els valors del nombre d'arbres [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150].
2. Seguidament hem entrenat i predit els corresponents subsets 5 cops variant el valor del nombre d'arbres mentre que manteníem constants la mida de cel·la, la mida de bloc i el nombre de contenidors. Es varen executar 5 cops degut a que la classificació te un factor random.

Els resultats obtinguts es poden observar a la Figura 15. Observem que el comportament del classificador és aproximadament el mateix per a diferents valors del nombre d'arbres. Tenim que quant més augmenta el nombre d'arbres més augmenta el temps computacional, per tant ens interessa agafar un valor que equilibri temps computacional i error obtingut. Hem agafat com millor valor del nombre d'arbres 90, ja que encara que el classificador amb aquest nombre d'arbres no dona el millor error, és més ràpid. El comportament creixent del TreeBagging s'observa amb més claredat a la Figura 15b.

Size	Accuracy	Looking	Not Looking
10	81.73%	8.75%	32.73%
20	83.94%	8.53%	27.49%
30	84.92%	8.89%	24.46%
40	85.16%	9.07%	23.58%
50	85.12%	9.00%	23.80%
60	85.01%	9.00%	24.08%
70	85.60%	9.15%	22.37%
80	85.43%	9.07%	22.92%
90	85.41%	9.07%	22.98%
100	85.76%	8.42%	23.09%
110	85.47%	8.86%	23.14%
120	85.71%	8.68%	22.81%
130	85.78%	8.75%	22.53%
140	85.78%	9.07%	22.04%
150	86.06%	8.53%	22.15%

(a) Taula amb els resultats de l'estudi de la mida de l'arbre.



(b) Gràfica amb els resultats de l'estudi de la mida de l'arbre.

Figura 15: Resultats de l'estudi de la mida de l'arbre.

6 Conclusions

Com hem pogut observar a les Seccions 4.2 i 5.2 es de vital importància l'ajust correcte dels paràmetres que van apareixent a l'hora de l'aprenentatge d'un classificador; ja que buscant els "millors" valors d'aquests podem millorar dràsticament la precisió de la classificació.

Aixi mateix, la qualitat de la predicció també esta condicionada en part per les dades escollides per entrenar, com hem vist a la Secció 4.2.4. Un bon treball pràctic posterior a aquest pot ser estudiar quines son les millors imatges per entrenar alhora de detectar mirades en fotografies.

Finalment cal comentar que ens hem trobat amb una gran dificultat a l'hora d'escollir les característiques pels diferents classificadors. La resposta a aquesta pregunta no la té ningú, i possiblement és el factor que més influeix en la correctesa del classificador.

Referències

[1] Paul Viola and Michael J Jones. "Robust real-time face detection". In: *International journal of computer vision* 57.2 (2004), pp. 137–154.

[2] Robert K McConnell. *Method of and apparatus for pattern recognition*. US Patent 4,567,610. Jan. 1986.

[3] *Extract histogram of oriented gradients (HOG) features: Cell size*. URL: https://es.mathworks.com/help/vision/ref/extracthogfeatures.html?requestedDomain=true#input_argument_namevalue_d119e131609.

[4] *Extract histogram of oriented gradients (HOG) features: Block size*. URL: https://es.mathworks.com/help/vision/ref/extracthogfeatures.html?requestedDomain=true#input_argument_namevalue_d119e131634.

[5] Hough Paul VC. *Method and means for recognizing complex patterns*. US Patent 3,069,654. Dec. 1962.