Universitat Autònoma de Barcelona

escola d'enginyeria

4138 - APLICACIÓ DE VISIÓ PER A DISPOSITIUS MÒBILS

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Marc Fernández Gironès
i dirigit per
Jordi González Sabater
Bellaterra, 20 de juny de 2011

Universitat Autònoma de Barcelona

etse

Escola Tècnica Superior d'Enginyeria

El sotasignat, Jordi González Sabater

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Marc Fernández Gironès

I per tal que consti firma la present.

P.O

Signat: .....

Bellaterra, 20 de juny de 2011

TAULA DE CONTINGUTS

CAPÍTOL 1: INTRODUCCIÓ

1.1. OBJECTIUS

L'objectiu d'aquest projecte és l'estudi i la creació d'una aplicació de visió per computador que funcioni pels dispositius mòbils.

Els dispositius mòbils que farem servir són qualsevol versió de l'iPhone d'Apple ja que disposen de càmera i un processador suficientment potent com per córrer aplicacions que tractaran amb imatges sense cap tipus de problema. Una altra opció seria usar els últims dispositius Android, els quals també són suficientment potents. De totes maneres, hem escollit els iPhone perquè són els *smartphones* més utilitzats i llavors la nostra aplicació podrà ser descarregada per més persones.

Des del punt de vista del desenvolupament, l'aplicació tindrà els següents funcions:

- Detecció de cares.
- Tractament de les imatges.
- Reconeixement de gènere.
- Reconeixement facial.

Tot això s'implementarà amb les llibreries OpenCV[1], les quals amb una compilació creuada, serem capaços d'executar-les en els processadors dels dispositius d'Apple.

Des del punt de vista de l'usuari final, l'aplicació demanarà un fotografia d'una persona, ja sigui extreta del àlbum del dispositiu mòbil o feta amb la càmera, i després de 2 passos i de detectar el gènere, retornarà la persona famosa que més s'assembla a la fotografia inicial.

[1] OpenCV (Open Source Computer Vision Library): Conjunt de funcions dirigides principalment per la visió per computador en temps real, desenvolupat per Intel i ara amb el suport de Willow Garage.

1.2. MOTIVACIONS

Les motivacions que em van portar a fer aquest projecte van ser la voluntat d'aprendre a programar i fer aplicacions per els dispositius mòbils i en particular per l'iPhone ja que és un negoci que està a l'alça.

Per una altra banda, l'absència de bones aplicacions de reconeixement facial i el descontentament de la gent amb elles em van empènyer a plantejar una aplicació per tal de cobrir els buits que hi havia en aquell moment en aquest àmbit.

A més a més, la visió per computador és unes de les branques que més m'agraden de la informàtica, i així vaig deixar patent també fent el projecte de final de carrera de la Enginyeria Tècnica de Sistemes al 2008. Aquest projecte és, en part, la continuació d'aquell que consistia en el reconeixement de postures de les cares i es va fer amb Matlab, utilitzant uns altres algoritmes de visió per computador i aprenentatge automàtic.

Tot i que a la carrera d'Enginyeria Informàtica es tracten temes de visió per computador i intel·ligència artificial, la durada de les assignatures sol ser quadrimestral i per tant no dóna temps a aprofundir en la matèria. Amb aquest projecte vull entrar en més detall en aquells algoritmes que ens van explicar per sobre i ajuntar-ho tot en una aplicació que tingui una funció especifica.

Finalment, vaig entendre que la millor forma d' aprendre a programar per dispositius mòbils era auto obligant-me ja que per motius de feina i falta de temps mai havia tingut l'oportunitat. Llavors, escollint un projecte d'aquestes característiques no em quedava un altre remei que aprendre els fonaments de la programació per mòbils i particularment per l'iPhone.

1.3. ESTRUCTURA DE LA MEMÒRIA

Aquesta memòria està dividida en 8 capítols que són els següents:

- **Capítol 1:** En el primer dels capítols tenim els objectius, on expliquem breument i clarament les intencions d'aquest projecte. També podem trobar les motivacions per les quals hem desenvolupat aquest projecte, l'estructura de la memòria, una petita introducció a la visió per computador, l'estat de l'art i la nostra proposta de solució.

- **Capítol 2:** En el segon capítol trobem l'estudi de viabilitat fet per comprovar que el projecte era factible de fer i també la seva planificació juntament amb unes conclusions.

- **Capítol 3:** En el capítol tres entrem més en detall en les especificacions de l'aplicació i trobem els requisits funcionals, no funcionals i desitjables del sistema juntament amb els usuaris que participen i un diagrama de casos d'ús.

- **Capítol 4:** El quart capítol es centra en el disseny del software pel que fa els diagrames de classes a utilitzar i quina serà l'arquitectura del nostre software.

- **Capítol 5:** En el cinquè capítol trobem l'explicació detallada de les metodologies utilitzades per a la implementació de l'aplicació amb els algoritmes i funcions que hem fet servir.

- **Capítol 6:** El capítol número sis expliquem els resultats que hem obtingut després d'utilitzar les metodologies explicades en l'apartat anterior.

- **Capítol 7:** En el penúltim capítol trobem les propostes de treball futur que nosaltres creiem que farien millorar els resultats detallats en el capítol sis.

- **Capítol 8:** El darrer capítol conté les conclusions extretes a partir de fer aquest projecte amb les metodologies aplicades i els resultats obtinguts

1.4. VISIÓ PER COMPUTADOR

La visió per computador, també coneguda com visió artificial o visió tècnica, és un subcamp de la intel·ligència artificial. El seu propòsit és programar un computador per a que "entengui" una escena o les característiques d'una imatge. A la Figura 1 [1] podem observar les diferents àrees de la visió per computador.

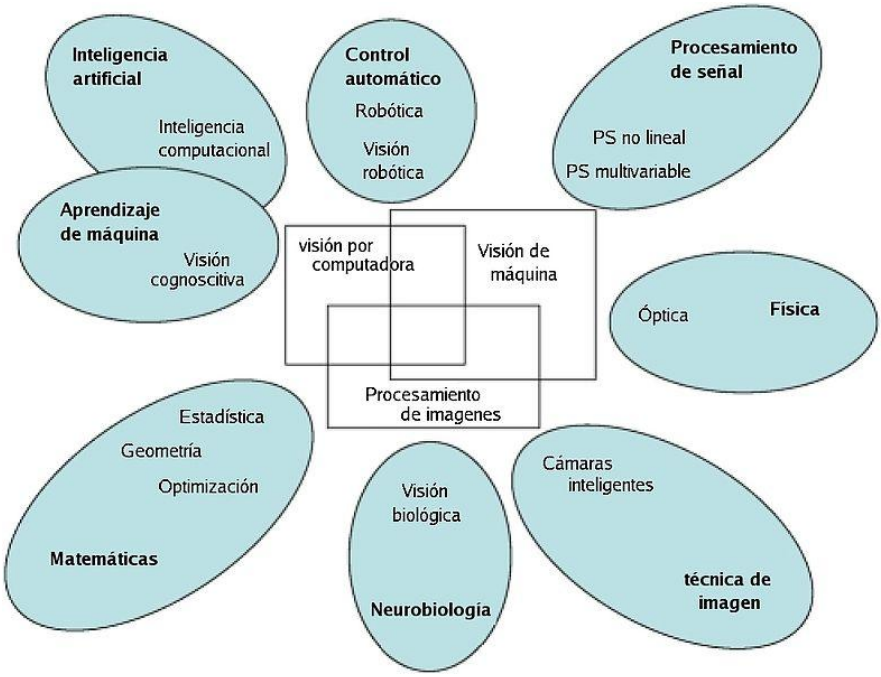


Figura 1: Esquema de relacions entre Visió per Computador i altres àrees afins.

Alguns exemples de branques de la visió per computador serien:

- **Reconeixement:** Es reconeixen un o varis objectes pre-especificats o "apresos".
- **Identificació:** Es reconeix una instància individual d'un objecte. Per exemple la identificació d'un vehicle específic.
- **Detecció:** On les dades de la imatge s'exploren per una condició específica. Per exemple la detecció de possibles cèl·lules o teixits anormals en imatges mèdiques.
- **Moviment:** Seguiment dels moviments d'un objecte.
- **Reconstrucció de l'escena:** Donades una o vàries imatges d'una escena o un vídeo es fa un model en 3D de l'escena.
- **Restauració de la imatge:** Eliminar el soroll de les imatges.

Un sistema típic de visió per computador està format per les següents funcions:

- **Adquisició de la imatge:** On una càmera capta la imatge mitjançant sensors i la digitalitza.
- **Procés previ:** Normalment s'han de processar les dades per tal de realçar algunes característiques, com per exemple la reducció de soroll, l'augment del contrast o l'escalat.
- **Extracció de característiques:** Les característiques de la imatge s'extrauen de les dades. Els exemples típics d'aquestes característiques són: línies, cantonades i algunes altres que es poden relacionar amb la textura o la forma.
- **Detecció/Segmentació:** En aquest punt es pren la decisió sobre quins són els punts o les regions de la imatge que són rellevants per la posterior transformació.
- **Procés d'alt nivell:** En aquest pas l'entrada és un sistema de dades (un sistema de punts o una regió) i es tracta d'estudiar aquestes dades per veure si compleixen alguna condició preestablerta o es classifiquen, etc.

Aquests punts anteriors són les passes que es seguiran en aquest projecte per tal de dur-lo a terme.

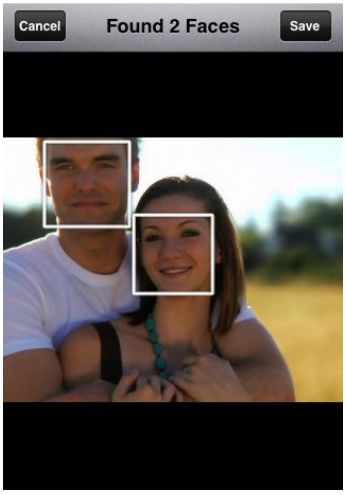
1.5. ESTAT DE L'ART

Actualment existeixen tres diferents aplicacions a la *App Store*[2] que fan algun tipus de reconeixement de cares i les quals mostrem a continuació en un ampli estudi de les característiques de cadascuna de elles.

FACE MATCH

Face Match és una aplicació creada per Polar Bear Farm Ltd. que detecta cares i com ells diuen a la descripció de iTunes, "intenta" reconèixer cares en la foto.

Utilitzen algun tipus d'algoritme d'aprenentatge perquè diuen que aprèn cada cop que tu poses el nom a una foto. L'inconvenient és que les imatges són transferides des de l'iPhone als seus servidor per fer la detecció i el reconeixement de les cares així que requereix de connexió d'internet per funcionar.

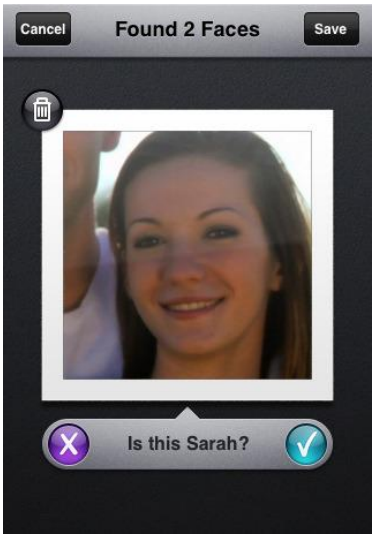


La integració social amb Facebook funciona bé, adquireix tant els teus contactes de la xarxa social com els de la teva agenda per mostrar-te una auto completament quan estàs escrivint els nom d'una cara no reconeguda.

La categoria de l'aplicació és Fotografia, l'última actualització data del 13 de juliol del 2010, està testejada en iOS 4.0, té un pes de 3.4Mb i un preu de $4.99.

Les puntuacions que els clients han donat a aquesta *app* no són bones amb una mitjana de dues estrelles. En total té 259 puntuacions de les quals 130 són només d'una estrella. A continuació mostrem algunes de les ressenyes originals sense traduir de la tenda d'Estats Units.

[2] App Store: És una plataforma digital de distribució d'aplicacions per els sistemes Apple.



"Doesn't work at all (1 out of 5)

by Marlins2003

This app has so many issues it uploaded photos with the same name of different persons in my profile creating a new photo album. It doesn't create Face Recognition faces by pulling the profiles of these persons to find if there face have changed. It even changed my password on Facebook profile and uploaded these photos without my authorization. The connection to Facebook it would only say it could not connect fix these issues before creating an application that says definite face recognition."

"bad (1 out of 5)

by zatchsterza1994

This is a froud and over priced app that hard to use import takes forever DO NOT GET THIS APP OR YOU'll be throwing away money"

"Will NOT WORK (1 out of 5)

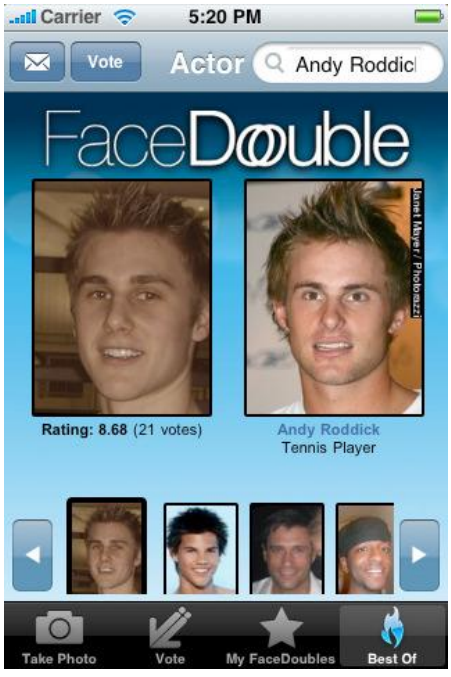
by speedyboy94

This app will not work at all!! Waste of money!!!!!!"

FACEDOUBLE CELEBRITY LOOK ALIKE

FaceDouble per FaceDouble Inc. vindria a ser la mateixa idea que volem implementar: Un emparellament amb persones famoses de les fotos fetes des del telèfon. Ells diuen que l'aplicació fa emparellament de cares i compara amb milers de persones famoses d'avui dia. També tenen una bona integració amb Facebook ja que els teus amics de la xarxa social poden donar el seu vistiplau del reconeixement. El preu és més raonable, $1.99. Està categoritzada com "Estil de Vida" i l'última actualització va ser el 9 d'abril del 2010.

Sembla ser que tenen bastant problemes per detectar els ulls a les imatges i per tant no poden fer el reconeixement/emparellament amb la persona famosa. Per aquesta raó han escrit algunes recomanacions a la descripció de l'iTunes per ajudar als usuaris a fer-la servir. Algunes d'aquestes recomanacions són que la fotografia sigui totalment frontal, que tinguis la millor il·luminació possible i que no portis ulleres a la foto. Malgrat aquestes recomanacions, tenen una molt mala puntuació de dues estrelles de 5 i tenen una estrella en 723 ocasions sobre un total de 1313. A continuació tenim alguns comentaris originals d'usuaris que han descarregat aquesta aplicació en l'App Store americana:



"Always picks same celebs. Looks nothing like the person you're photo... (1 out of 5)
by fprevin

Nice idea, but just doesn't work. First, it tells you the person looks like a celebrity who looks nothing like that person. Use it a bit more, and you find out there's a very small pool of celebrity faces it chooses from."

"Stupid app (1 out of 5)
by NeffyFeff

This is so lame. It won't take any of my photos. I want my money back."

"Aparently I have no eyes!! (1 out of 5)
by Lindzymae

Every picture tried, it said 'eyes could not be found'. I would like to have my $$ back."

FACELOOK FACE RECOGNITION

FaceLook es descriu com l'única aplicació de reconeixement de cares basada en Facebook. El que fa és simple: tu fas una fotografia i l'emparella amb un dels teus amics de la xarxa social i et diu el seu nom i tots els seus detalls. Per fer això, l'aplicació necessitarà connexió a internet i estar "*loguejat"* a Facebook.

Aquesta *app* està feta per Nir Marcus i em va dir que utilitza els algoritmes de Face.com per fer el reconeixement. Face.com es una companyia tecnològica que treballa amb reconeixement de cares i que ofereix una plataforma per desenvolupadors per detectar i reconèixer cares en fotos automàticament amb una API REST gratuïta, que permet integració instantània amb xarxes socials, llocs web de compartició de fotografies, motors de cerca i més. Per a la detecció i el reconeixement utilitzen els seus propis algoritmes i no OpenCV com en principi farem nosaltres.

El tipus de l'aplicació és "Entreteniment", l'última actualització va ser el 2 de novembre de 2010, té un pes

de 0.7Mb i el preu és de $1.99 per 300 reconeixements. Quan tu acabes aquests 300 emparellaments, pots comprar més al preu de: 25 captures per $0.99, 60 captures per $1.99 i 100 captures per $2.99.

La seva puntuació no és gaire favorable tampoc, amb una mitja de dues estrelles sobre cinc i tenen una estrella en 65 comentaris en un total de 104. Les ressenyes mostren un comentari dolent en relació als paquets extra que has de comprar i dos bons parlant sobre el bon funcionament de l'aplicació:

"Rip Off!... (1 out of 5)
by Todd Michael

I was barely convinced to download this app for $1.99. Then AFTER I download the app tells me that I have to PAY for each face recognition! Yeah you get some free that come with the app but once those are gone you have to pay $1.00 for 25 more recognitions! Shame shame shame on you tricking me... I want my money back. You will get no more from me."

"Great app (5 out of 5)
by Duduyy

It actually works! I love it"

"Great app! (5 out of 5)
by Lindzymae

Amazing tool for face recognition. It recognizes people accurately. The application has 300 initial captures and if you want more you need to purchase."

Conclusió

Com a conclusió, la nostra aplicació hauria de ser forta en el reconeixement i sobretot en la detecció d'ulls per no caure en errors com els hi passa a les altres aplicacions. També haurem de tenir una àmplia base de dades de famosos i una bona integració amb xarxes socials. El preu haurà de ser competitiu i no seria mala idea algun tipus de model de paquets extres. Els nostres punts forts seran que no necessitarem internet per utilitzar l'aplicació i així es podrà fer servir en tots els entorns.

1.6. DISSENY DEL PRODUCTE

En aquest apartat fem un estudi per saber quin és el millor producte a desenvolupar per tal de cobrir el forat que hi ha ara mateix al mercat pel que fa aplicacions per iPhone amb un bon reconeixement de cares tal com hem vist a l'estudi de l'estat de l'art.

Necessitats del Consumidor

El consumidor necessita una aplicació portàtil per tal de fer imatges sobre la marxa i trobar el seu doble en aquell mateix moment i amb la possibilitat de compartir-ho amb els amics.

Intenció del Producte

La intenció del producte és entretenir i oferir al client una forma fàcil i divertida per descobrir nous dobles de la gent. De fet, vist les aplicacions que hi han ara mateix al mercat, volem crear una aplicació de dobles que sigui fiable.

Estructura de Funcions

En aquest subapartat estudiem quina seria la funció principal de la nostra aplicació i les subfuncions generals.

Funció primària: Trobar el doble d'una persona.

Subfuncions: Detecció de cares, detecció d'ulls, reconeixement de gènere, classificació de cares, compartició dels resultats.

Matriu Morfològica

A partir de les subfuncions detectades a l'apartat anterior, identifiquem les possibles solucions particulars per a cada subfunció.

Subfuncions / Solució	Solució 1	Solució 2	Solució 3
Detecció de Cares	Haar-Like Features	Detecció Manual	Crear algoritme propi
Detecció d'ulls	Haar-Like Features	Detecció Manual	Crear algoritme propi
Reconeixement de Gènere	Nearest Neighbor	Support Vector Machine	Xarxes Neurals
Classificació de cares	EigenFaces (PCA)	FisherFaces (LDA)	Kernel Methods
Compartició Resultats	Correu Electrònic	Xarxes Socials	Web pròpia

Taula I: Diferents solucions per cada subfunció.

Conceptes de Productes

Amb totes les solucions a les subfuncions, les combinem per tal de crear diferents solucions globals de com implementar el producte.

a) Haar Like Features – Haar Like Features – Nearest Neighbor – EigenFaces – Web pròpia.

b) Detecció pròpia – Detecció pròpia – Nearest Neighbor – EigenFaces – Web pròpia.

c) Crear algoritme propi – Crear algoritme propi – Nearest Neighbor – EigenFaces – Web pròpia.

d) Haar Like Features – Haar Like Features – Nearest Neighbor – EigenFaces – Social Networks.

e) Haar Like Features – Haar Like Features – Support Vector Machine – EigenFaces – Social Networks.

f) Haar Like Features – Haar Like Features – Xarxes Neurals – EigenFaces – Social Networks.

g) Haar Like Features – Haar Like Features – Nearest Neighbor – FisherFaces – Social Networks.

Criteris de Selecció

Avaluem cada proposta de producte de l'apartat anterior amb els següents criteris de selecció: Complexitat, usabilitat, cost, rendiment, eficàcia i temps de desenvolupament; i els resultats els podem veure a la Taula II. (Puntuacions basades en l'experiència i pàgines web especialitzades. Valors que van de l'1 al 5 on 1 és malament i 5 és bo).

Criteri / Solucions	Solució A	Solució B	Solució C	Solució D	Solució E	Solució F	Solució G
Complexitat	3	4	2	3	2	1	1
Usabilitat	2	1	2	4	4	4	4
Cost	3	3	1	4	4	4	4
Rendiment	4	4	3	4	4	4	4
Eficàcia	3	4	4	3	4	5	5
Temps de Desenvolupament	2	5	1	3	2	1	1
Total	**17**	**17**	**13**	**21**	**20**	**19**	**19**

Taula II: Comparació de les diferents solucions amb els diferents criteris de selecció.

Dissenys Viables

La solució D estava formada per detecció de cares i ulls mitjançant *Haar-Like Features*, el reconeixement de gènere amb *Nearest Neighbor*, la classificació de cares mitjançant *EigenFaces* i la compartició dels resultats mitjançant les xarxes socials. Aquesta solució és la que més puntuació ha obtingut en la nostra matriu morfològica ja que té un bon equilibri entre complexitat i efectivitat i és per tant la que durem a terme.

CAPÍTOL 2: ESTUDI DE VIABILITAT

2.1. RECURSOS NECESSARIS

Per realitzar el projecte implementant la solució escollida en l'apartat anterior necessitarem els següents recursos tant materials com humans:

Recursos software:

- Mac OS X versió 10.6.x
- XCode 4
- Interface Builder
- Instruments
- iPhone SDK
- iWork 09
- Projector 2.3
- OmniGraffle

Recursos mínims hardware:

- Processador Inter Core 2 Due a 2.0 GHz
- 2 Gb de memòria RAM
- iPhone

Recursos humans:

- Director de projecte
- Enginyer de software
- Dissenyadors gràfics

2.2. ANÀLISI DE COSTS

En la Taula III podem observar la relació de costos dels diferents materials i software que necessitaríem per realitzar el projecte.

Recurs	Cost Total	Cost Mensual
Amortització Ordinador Programació	€700	
Amortització XCode 4	€3.99	
Amortització iWork	€60	
Amortització Projector	€34.50	
Amortització OmniGraffle	€69	
Total	**€ 867.49**	**€0**

Taula III: Relació de costos materials.

A la Taula IV veiem un sou per hora aproximat per a cadascunes de les persones necessàries per realitzar el projecte.

Recurs	Cost per hores
Director de projecte i programador	
Marc Fernandez	5€/h
Dissenyadors Gràfics	
Anna Sanz	4€/h
Jordi Puig	4€/h

Taula IV: Relació de costos personals.

Finalment podem veure a la Taula V el cost que suposen cadascuna de les tasques tenint en compte els sous aproximats dels recursos humans abans esmentats.

Tasques	Recursos Assignats	Cost
Market Research	Marc Fernandez	€780.00
iPhone Development Learning	Marc Fernandez	€5,580.00
Learning Algorithms	Marc Fernandez	€2,280.00

Tasques	Recursos Assignats	Cost
Viability Study	Marc Fernandez	€900.00
Bussines Plan	Marc Fernandez	€1,200.00
Port OpenCV libraries into iPhone	Marc Fernandez	€780.00
Image Processing	Marc Fernandez	€1,200.00
Application Implementation	Marc Fernandez	€7,140.00
Train Learning Algorithms	Marc Fernandez	€5,220.00
Build Starting Database	Marc Fernandez	€1,320.00
Logo Design	Anna Sanz	€240.00
Application Design	Marc Fernandez, Jordi Puig	€756.00
Report	Marc Fernandez	€3,000.00
Website Design	Marc Fernandez	€1,080.00
TOTAL		**€ 31,476.00**

Taula V: Relació de tasques amb recursos assignats i costos més el cost total per tasques del projecte.

2.3. AVALUACIÓ DE RISCOS

En aquest projecte, bàsicament existeixen dos riscos importants. El primer d'ells i el més important i que impediria la realització del mateix és que no puguem portar les llibreries OpenCV a l'entorn iPhone. Per això caldria fer un bon estudi previ i en cas de no poder fer-ho cancel·laríem el projecte el més aviat possible per evitar pèrdues innecessàries.

Un altre risc seria que l'algoritme d'aprenentatge sigui poc robust i no aconsegueixi classificar/emparellar les fotografies de les persones amb els famosos. Si fos aquest el cas, buscaríem un altre algoritme de classificació més adequat.

2.4. VIABILITAT LEGAL

Per la realització del projecte presentat ens haurem de regir a la Llei Orgànica 1/1982, del 5 de maig, de Protecció Civil del Dret a l'Honor, a la Intimitat Personal i Familiar i a la Pròpia Imatge. Exactament al capítol setè cinc, el qual prohibeix:

"La captación, reproducción o publicación por fotografía, filme, o cualquier otro procedimiento, de la imagen de una persona en lugares o momentos de su vida privada o fuera de ellos."

Per aquest motiu, a totes les persones que han col·laborat per aquest projecte amb la seva imatge se les ha fet signar un document que ens autoritzen a poder mostrar fotografies seves i a més a més donen aquestes imatges a la ciència per ser utilitzades en bases de dades per l'estudi.

També ens haurem de regir per la Llei del Reial Decret Legislatiu 1/1996, del 12 d'abril, de la Propietat Intel·lectual la qual en el seu primer article diu que:

"La propiedad intelectual de una obra literaria, artística o científica corresponde al autor por el solo hecho de su creación."

Així que per mostrar fotografies extretes d'internet de famosos a la nostra aplicació haurem de regir-nos per el Copyright[3] de la imatge i demanar permís per el seu ús o citar a l'autor.

[3] Copyright - Forma de protecció proporcionada per les lleis vigents a la majoria dels països que dona al propietari d'una obra el dret exclusiu per a fer i per a autoritzar a altres l'ús d'aquesta.

2.5. PLANIFICACIÓ

Per fer la planificació, primer hem dividit el projecte en les següents tasques:

1. Estudi del mercat. Mirarem quines aplicacions hi han al mercat sobre el tema que volem tractar.
2. Aprenentatge del desenvolupament per iPhone. Farem uns primers tutorials en la programació per iPhone per tal d'aprendre la nova sintaxi i les diferents funcions especifiques del sistema.
3. Implementació dels algoritmes d'aprenentatge. Desenvolupament dels algoritmes escollits per tal de fer tot el procés de reconeixement de cares.
4. Estudi de viabilitat.
5. Pla d'empresa. Estudi de l'informe de mercat i presa decisions sobre el disseny del producte.
6. Compilació creuada de les llibreries OpenCV per l'iPhone
7. Processament de les imatges. Creació de tots els algoritmes per el processament previ de les imatges.
8. Implementació de l'aplicació en l'entorn iPhone.
9. Entrenament dels algoritmes d'aprenentatge.
10. Creació de la base de dades. Adquisició de les imatges dels famosos.
11. Disseny del logotip.
12. Disseny de la interfície de l'aplicació.
13. Creació de la memòria.
14. Disseny de la pàgina web.

També hem identificat dos *milestones*[4] que són l'entrega de la memòria per la setmana del 17 al 22 de juny i la presentació al tribunal del 27 de juny al 8 de juliol. A partir d'aquestes dates hem elaborat la planificació temporal del nostre projecte del qual podeu veure a continuació el diagrama de Gantt (Figura 2) i les tasques en forma de taula (Taula VI) amb informació més detallada.

[4] Milestone - Final d'una etapa que marca l'acompliment d'una feina o fase.

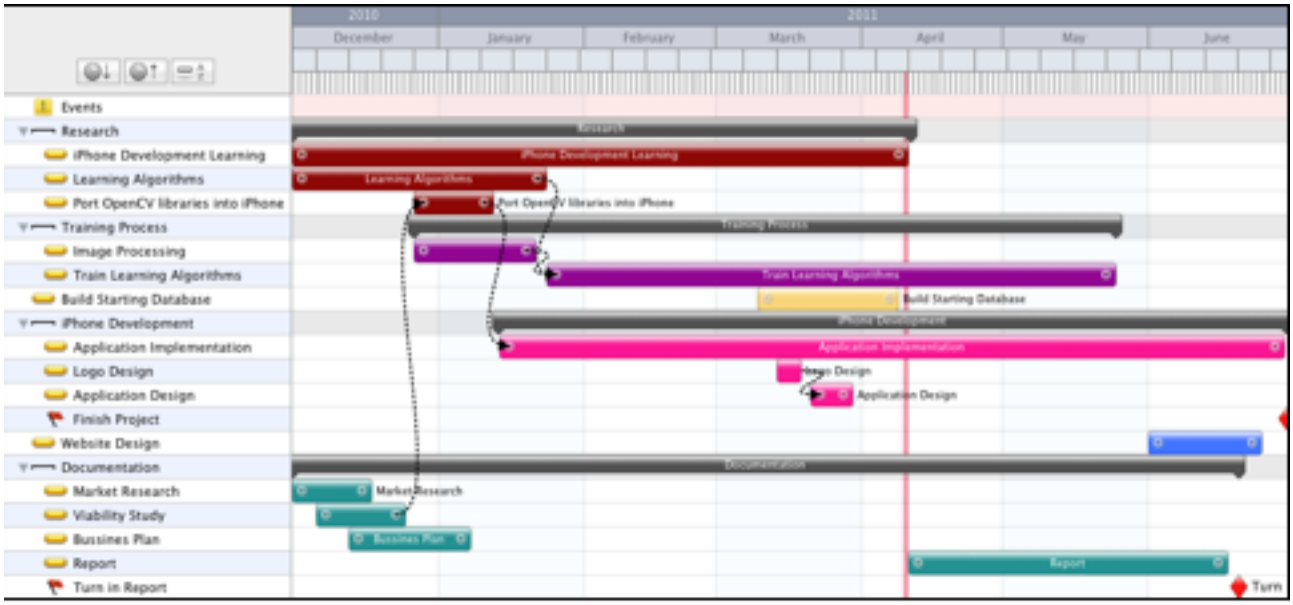


Figura 2: Diagrama de Gantt de la planificació del projecte.

Al diagrama de Gantt podem veure com vam començar el projecte al desembre del 2010 fent un aprenentatge sobre el desenvolupament de l'iPhone i començant a pensar i a implementar els algoritmes d'aprenentatge.

Id	Tasques	Inici	Fi	Estimació (dies)	Duració Aprox. (dies)	Predecessora
1	Market Research	1-Dec-2010	17-Dec-2010	13	10	
2	iPhone Development Learning	1-Dec-2010	11-Apr-2011	93	100	
3	Learning Algorithms	1-Dec-2010	24-Jan-2011	38	38	
4	Viability Study	6-Dec-2010	24-Dec-2010	15	15	
5	Bussines Plan	13-Dec-2010	7-Jan-2011	20	20	
6	Port OpenCV libraries into iPhone	27-Dec-2010	12-Jan-2011	13	30	4
7	Image Processing	27-Dec-2010	21-Jan-2011	20	20	
8	Application Implementation	14-Jan-2011	29-Jun-2011	119	102	6
9	Train Learning Algorithms	24-Jan-2011	25-May-2011	87	87	3, 7
10	Build Starting Database	10-Mar-2011	8-Apr-2011	22	22	
11	Logo Design	14-Mar-2011	18-Mar-2011	5	5	
12	Application Design	21-Mar-2011	30-Mar-2011	7	7	11
13	Report	11-Apr-2011	17-Jun-2011	50	50	
14	Website Design	1-Jun-2011	24-Jun-2011	18	18	

Taula VI: Llistat de tasques amb dates d'inici, fi, duració prevista i real i relació entre tasques.

2.6. CONCLUSIONS

A partir d'aquest document podem extreure com a conclusions que pel que fa l'estat de l'art avui en dia, sí és veritat que existeixen aplicacions més o menys semblants a la que volem implementar en el nostre projecte però són de poca qualitat, tal i com diuen explícitament els comentaris.

Llavors, per ser millors que aquestes aplicacions, hem de no cometre els mateixos errors que elles i per exemple fer una bona classificació de gènere que ens donarà més opcions després per aproximar la persona d'una fotografia a un famós. També hem de ser forts en la detecció d'ulls ja que és primordial pel posterior reconeixement tal i com els hi passa a l'equip de FaceDouble i finalment necessitem una correcta integració amb les xarxes socials per córrer la veu del nostre producte i donar-lo a conèixer a més gent.

Per fer tot això hem identificat detalladament diferents tasques que són essencials juntament amb els recursos necessaris per fer cadascuna d'elles i amb una correcta planificació tindrem el producte llest per ser llençat al mercat a principis/mitjans de juliol.

Un últim factor a tenir en compte abans del llançament del producte i que pot condicionar tot el projecte és les normes a les quals ens hem de regir per tal que no tinguem un incompliment de les lleis que retardaria considerablement la sortida de l'aplicació al mercat.

CAPÍTOL 3: REQUERIMENTS

Aquest capítol tracta sobre les funcionalitats que ha de tenir la nostra aplicació en termes de requeriments funcionals, no funcionals i tipus d'usuaris.

3.1. CASOS D'ÚS

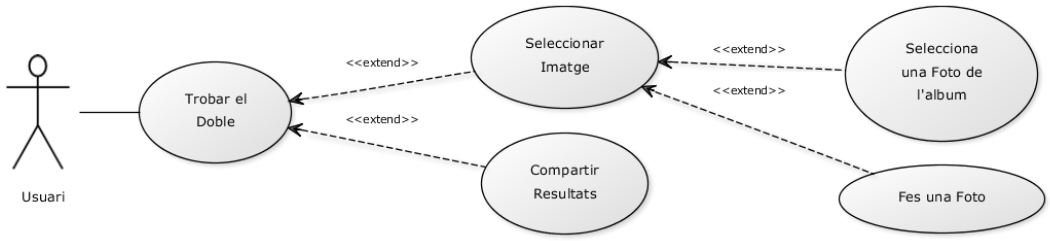


Figura 3: Diagrama de casos d'ús de l'aplicació.

A la Figura 3 podem observar l'únic cas d'ús de la nostra aplicació que consta d'un únic usuari el qual l'acció que portarà a terme és la de trobar el doble d'una imatge provinent de la càmera o que tenia prèviament guardada a l'àlbum del dispositiu.

3.2. TIPUS D'USUARIS

La nostra aplicació només tindrà un tipus d'usuari que interactuarà amb l'aplicació i aquest serà el client final. El client descarregarà l'aplicació i sense cap tipus de sistema d'autenticació podrà començar a utilitzar-la. Es farà una foto a ell mateix o algú altre i el sistema retornarà la imatge del personatge famós al que s'assembla.

3.3. REQUERIMENTS DEL SISTEMA

En el següent apartat enumerem tots els requeriments del sistema separats per requisits funcionals, requisits no funcionals i requisits desitjables.

3.3.1. REQUERIMENTS FUNCIONALS

RF1: L'aplicació ha de permetre a l'usuari escollir una fotografia de l'àlbum del seu telèfon.

RF2: L'aplicació ha de permetre a l'usuari fer una fotografia utilitzant la càmera integrada del dispositiu mòbil.

RF3: Un cop escollida una fotografia per l'usuari, aquesta s'ha de mostrar amb la cara detectada i donar l'opció a l'usuari de continuar amb aquesta o escollir una altra.

RF4: Un cop escollida una fotografia per l'usuari, es mostrarà una frase de cortesia que serà especifica per cada gènere.

RF5: Qual l'usuari accepta continuar el procés amb la fotografia escollida, el sistema mostrarà la seva imatge un altre cop juntament amb la imatge del personatge famós al que s'assembla.

3.3.2. REQUERIMENTS NO FUNCIONALS

RNF1: L'aplicació ha de tenir un disseny atractiu i únic.

RNF2: L'aplicació no hauria d'utilitzar internet per tal de trobar el doble de l'usuari.

RNF3: L'aplicació ha de trobar el doble de l'usuari en un temps raonable.

3.3.3. REQUERIMENTS DESITJABLES

RD1: L'usuari podrà publicar en el seu mur de Facebook el resultat del doble trobat.

RD2: L'usuari podrà publicar al seu Twitter el resultat del doble trobat.

RD3: Pàgina web explicativa de les característiques de l'aplicació i de suport.

3.4. LLISTA DE PRIORITATS DELS REQUERIMENTS

En aquest apartat llistem tots els requeriments del sistema identificats en l'apartat anterior per ordre de prioritat on els elements amb P1 són els més prioritaris i els elements amb P3 són els que són menys importants.

3.4.1. REQUERIMENTS AMB PRIORITAT ALTA (P1)

Els requeriments amb prioritat alta són: RF1, RF2, RF3, RF5, RNF2 i RNF3.

3.4.2. REQUERIMENTS AMB PRIORITAT MITJA (P2)

Els requeriments amb prioritat mitja són: RF4, RNF1.

3.4.3. REQUERIMENTS AMB PRIORITAT BAIXA (P3)

Els requeriments amb prioritat baixa són: RD1 i RD2.

CAPÍTOL 4: DISSENY DEL SOFTWARE

En aquest capítol explica com és l'arquitectura de l'iPhone, també conté informació sobre les classes utilitzades per tal d'implementar l'aplicació, diagrames de disseny i el disseny gràfic realitzat.

4.1. ARQUITECTURA DE L'IPHONE

L'arquitectura de l'iPhone és un disseny molt simple en termes de la teoria del sistema operatiu. En la Figura 4 podem observar un resum del procés sencer de com el hardware interactua amb el software del dispositiu.

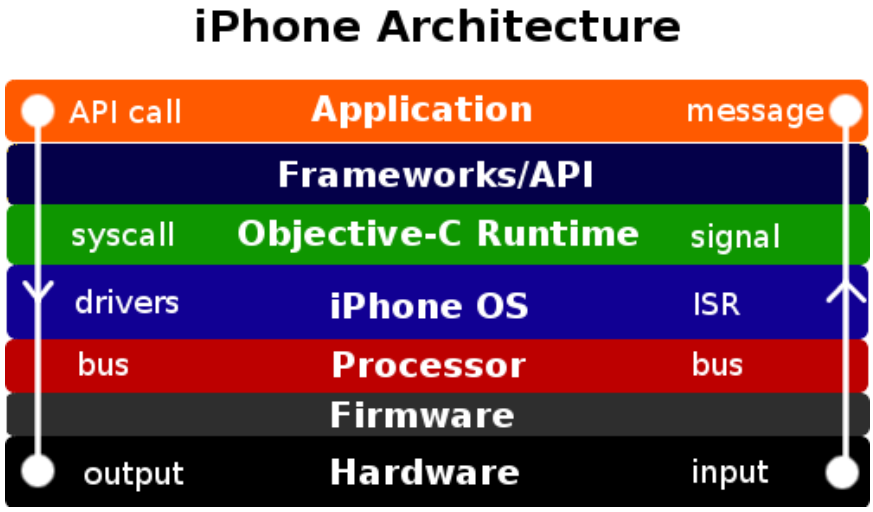


Figura 4: Diagrama de l'arquitectura de l'iPhone.

- **Application:** Aquesta és l'aplicació en execució. Aquesta aplicació ha estat compilada a codi natiu mitjançant el compilador distribuït per Apple i enllaçat amb el motor d'execució de l'Objective-C i les llibreries C amb l'enllaçador.

- **Framworks/API:** Aquí trobem el Cocoa Touch, les crides d'alt nivell a les llibreries OpenGL, etc. Aquestes crides a APIs són simplement capçaleres ja distribuïdes per Apple amb l'iPhone SDK, amb alguna cosa d'enllaçat dinàmic que té lloc en

temps d'execució. Aquesta resideix a sobre de la capa d'Objective-C ja que moltes de les APIs estan escrites en Objective-C. Aquí és, per exemple, on trobem les llibreries OpenCV que utilitzarem per tot el tractament d'imatges i reconeixement de cares.

- **Objective-C Runtime:** Aquesta capa està compresa per les llibreries d'Objective-C enllaçades dinàmicament i també per les llibreries subjacents de C. Les llibreries C són les que preparen l'entorn i per això les hem inclòs a la mateixa capa.

- **iPhone OS:** Aquest és el nucli, els controladors i serveis que formen part del sistema operatiu iPhone. Això a vegades es diu iPhone OS, iPhone OS X, o simplement OS X, però tot es refereix al mateix: Es posa entre l'espai de l'usuari i el hardware.

- **Processor:** Aquesta capa no es refereix al xip ARM, sinó que es refereix al conjunt d'instruccions ARM de baix nivell i la taula de descriptors d'interrupció tal com va ser creat per l'iPhone OS durant l'arrencada i la inicialització del controlador.

- **Firmware:** Aquesta capa fa referència al codi de xip específic que ja sigui amb la memòria continguda al voltant del perifèrics, o al controlador per a aquest perifèric (exemple: la pantalla tàctil o giroscopi).

- **Hardware:** Es refereix als chips físics soldats a la circuiteria de l'iPhone. Tot el que pots sentir o veure està en aquesta capa. El processador el trobem aquí i les seves instruccions, com ja hem comentat, a la capa de "Processor".

Compilació Creuada Llibreries OpenCV

Les llibreries OpenCV (OpenSource Computer Vision) són unes llibreries de funcions de programació per visió per computador en temps real. Van estar escrites originalment en C però tenen una completa interfície en C++ i tot el nou desenvolupament s'està fent en C++. Com que estan pensades per processadors

Intel tant en Windows o en Unix, per tal de fer-les funcionar en els chips ARM que porten incorporats els dispositius d'Apple, cal fer una compilació creuada mitjançant CMake per adaptar-les.

4.2. DIAGRAMA DE CLASSES

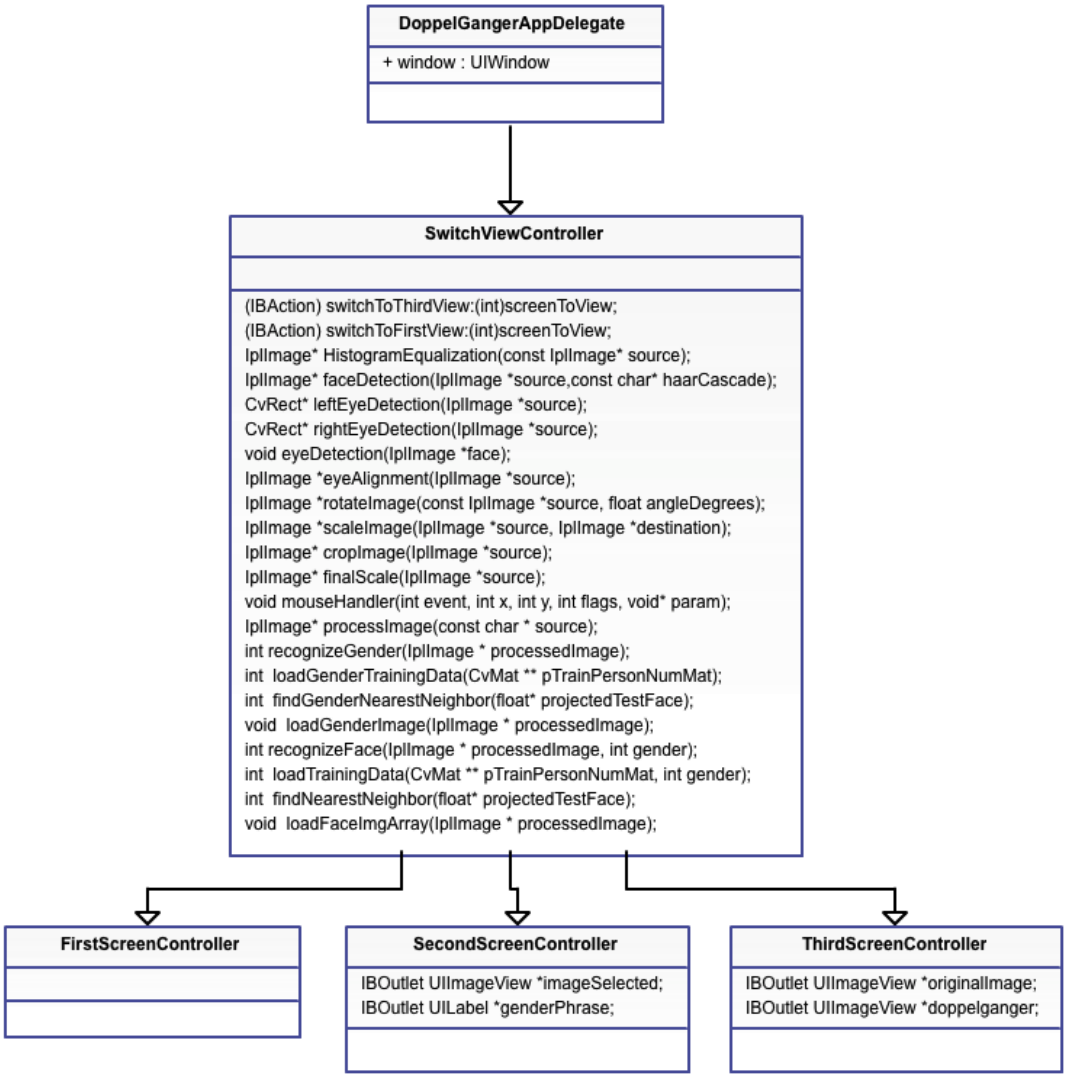


Figura 5: Diagrama de classes de la nostra aplicació.

Les classes que utilitzarem i que podem observar a la Figura 5 són la classe *DoopelGangerAppDelegate* que és la classe general del sistema i que és l'encarregada de carregar tots els frameworks[5] necessaris i de mostrar la UI.

[5] Framework: Conjunt de llibreries o classes reutilitzables per un sistema software.

Després tenim la classe *SwitchViewController* que és la més important de la nostra aplicació ja que és l'encarregada de fer totes les funcions de detecció i reconeixement de cares. Finalment tenim les tres classes: *FirstViewController, SecondViewController* i *ThirdViewController* que són les que s'encarreguen de mostrar les vistes de cada una de les pantalles de l'aplicació.

4.3. DIAGRAMA D'ACTIVITATS

Com ja hem vist en apartats anteriors, l'objectiu de l'aplicació es trobar el doble d'una persona i podem veure tot el flux i el diagrama d'activitats en la Figura 6.

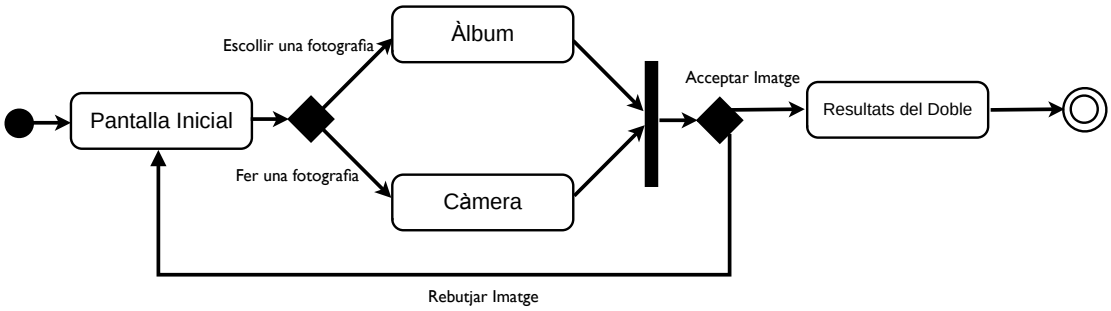


Figura 6: Diagrama d'activitats de la nostra aplicació.

Observem com l'usuari comença a la pantalla inicial on després podrà escollir entre seleccionar una fotografia des de l'àlbum del dispositiu o treure una foto amb la càmera. Un cop escollida la fotografia, tindrem l'opció de rebutjar-la i tornar al començament o d'acceptar-la i rebre els resultats del doble famós de la persona de la fotografia.

4.4. FLUX DE TREBALL

Un cop coneixem els passos a seguir per rebre els resultats del doble, podem fer un diagrama de flux amb el disseny prototip de l'aplicació (Figura 7)

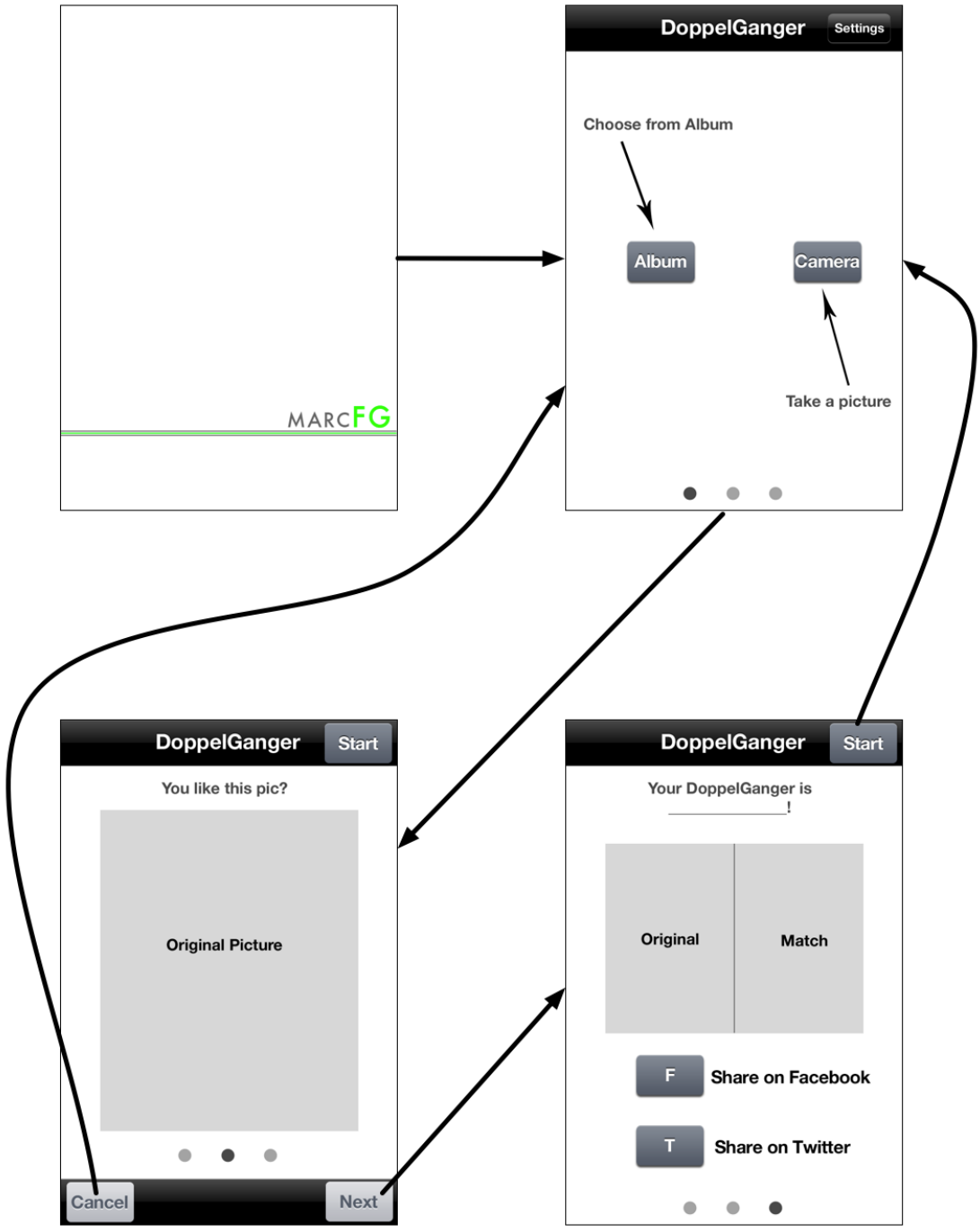


Figura 7: Diagrama de flux de la nostra aplicació.

Com ja vam comentar, l'aplicació solament tindrà tres passos i per tant només calen tres pantalles de les quals hem mostrat el seu disseny conceptual.

4.5. PROTOTIPS

En aquest últim apartat del disseny del software mostrem com serà la primera versió, i possiblement la definitiva, del disseny de cada pantalla de l'aplicació i del logotip.

4.5.1. LOGOTIP

El logotip ha estat dissenyat per la nostra companya Anna Sanz i podem veure el resultat a la Figura 8. Les lletres *DG* fan referència al nom de l'aplicació que serà *DoppelGanger.* Aquest nom té origen alemany i el que significa és doble fantasmagòric.

Figura 8: Logotip de l'aplicació.

4.5.2. PANTALLES DE L'APLICACIÓ

Figura 9: Disseny de les pantalles de l'aplicació.

A la Figura 9 tenim tots els dissenys de les pantalles en ordre d'esquerra a dreta i de dalt a baix. Estan igualment ordenats que el diagrama de flux i així podem comparar tots els elements. Aquests gràfics han estat fets pel nostre company Jordi Puig i per mi mateix.

CAPÍTOL 5: METODOLOGIA

Aquest capítol conté tota l'explicació detallada de tots els mètodes, algoritmes i tècniques que hem utilitzat per la implementació del projecte. Està dividit en quatre subapartats els quals expliquen cadascuna de les fases d'implementació de la nostra aplicació.

El primer apartat tracta sobre la detecció de cares mitjançant els classificadors en cascada anomenats *Haar-Like Features.* En el segon apartat, parlem del tractament que fem a les imatges per tal de preparar-les pel reconeixement i classificació. Al tercer tractem l'algoritme de l'Anàlisi de Components Principals que utilitzem tant en la classificació de gènere com en el reconeixement facial. Finalment, en l'últim apartat, expliquem com fem la classificació d'una nova instancia mitjançant l'algoritme de *K-Nearest Neighbor*.

5.1. DETECCIÓ DE CARES AMB HAAR-LIKE FEATURES

Per tal de fer la detecció de cares en una fotografia hem utilitzat el mètode *Haar-Like Features* [2] que ja ve implementat a les llibreries OpenCV i que va ser desenvolupat per Paul Viola [3] i millorat per Rainer Lienhart [4].

Aquest mètode va ser el primer framework de detecció d'objectes en proveir un competitiu percentatge de deteccions positives. Pot ser entrenat per detectar una gran varietat d'objectes però va ser primerament pensat per la detecció de cares.

El *Haar-Like Features* considera regions rectangulars adjacents en una localització especifica de la fotografia en una finestra de detecció, suma les intensitats dels píxels de les regions del mateix color i calcula la diferència entre elles. A la Figura 10 podem observar les diferents formes de calcular les *Haar-Like features* o característiques.

Cada característica és especificada per la seva forma, posició respecte la regió d'interès i l'escala. Per exemple, en el cas del 2c, la resposta és calculada com la diferència entre la suma de les intensitats dels píxels de tota la regió i la suma

dels píxels de la banda negra multiplicat per 3 per compensar per la diferència entre la mida de les àrees. La suma dels valors dels píxels en les regions rectangulars es calculen ràpidament utilitzant imatges integrals (veure Annex 1 – Imatges Integrals per més informació sobre aquest procediment).

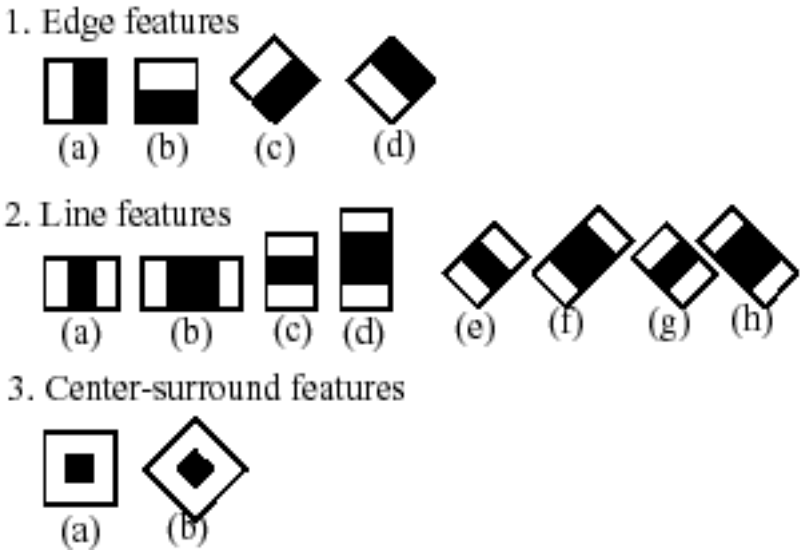


Figura 10: Diferents formes de calcular els Haar-Like features.

Després, aquesta diferència és utilitzada per categoritzar subregions de la imatge mitjançant un classificador en cascada. En cascada significa que el classificador resultant consisteix en varis classificadors dèbils[6] que són aplicats seqüencialment a la regió d'interès fins que alguna d'aquestes etapes dóna un resultat de detecció negatiu o bé totes les etapes són correctes i dóna un resultat de detecció positiu. Aquests classificadors, que per alguns objectes com per exemple cares ja venen inclosos amb les llibreries OpenCV, estan construïts mitjançant diferents tècniques de *boosting*[7] com per exemple el *Discrete AdaBoost*, *Real Adaboost*, *Gentle AdaBoost* o *Logitboost* [5].

Finalment, aquestes característiques calculades són l'entrada dels classificadors simples que són arbres de decisió amb, com a mínim, dues fulles dels

[6] Classificador dèbil: Quan el seu percentatge de classificació per a una tasca binària és una mica superior al 50%.

[7] Boosting: Meta-algoritme d'aprenentatge automàtic per realitzar aprenentatge supervisat.

quals obtindrem la classificació final que ens dirà si la regió que estem analitzant és o no l'objecte que estem buscant.

En resum, una finestra de dimensió i rotació canviants es va movent per la imatge i per cada subregió es calcula el *Haar-Like Feature.* Després la diferència es compara amb els llindars apresos (que ja estan proporcionats per les llibreries *OpenCV*) que separa els objectes, en el nostre cas cares, de les no-cares. Perquè cada *Haar-Like feature* és un classificador dèbil, es necessiten un gran nombre d'ells per tal de descriure un objecte amb suficient exactitud. Finalment, l'algoritme té de sortida un 1 si ha identificat algun objecte dels que estem buscant a la imatge o un 0 en cas contrari.

Les funcions proporcionades per les llibreries OpenCV que hem fet servir per realitzar tot aquest procés de la detecció de cares han estat:

- `void* cvLoad(const char* filename, CvMemStorage* storage=NULL, const char* name=NULL, const char** realName=NULL)`

Amb aquesta funció carreguem el classificador en cascada des d'un fitxer.

Paràmetres: - *filename*: Nom del fitxer que conté el classificador en cascada entrenat.
- *storage*: Emmagatzemament de memòria per estructures dinàmiques.
- *name*: Nom de l'objecte opcional.
- *realName*: Paràmetre de sortida opcional que contindrà el nom de l'objecte carregat.

- `CvSeq* cvHaarDetectObjects (const CvArr* image, CvHaarClassifierCascade* cascade, CvMemStorage* storage, doublescale_factor=1.1, int min_neighbors=3, int flags=0, CvSize min_size=cvSize(0, 0))`

Aquesta funció és la que detecta els objectes a la imatge i té els següents paràmetres:

Paràmetres:
- *image*: Imatge on vols detectar els objectes.
- *cascade*: Classificador en cascada prèviament carregat.
- *storage*: Emmagatzemament de memòria per guardar la seqüència de rectangles candidats a ser objecte.
- *scale_factor*: El factor per la qual la finestra de cerca serà re-escalada entre les subseqüents cerques.
- *min_neighbors*: Mínim nombre de rectangles veïns que faran un objecte.
- *flags*: Mode d'operació.
- *min_size*: Mida mínima de la finestra.

El classificador en cascada ja entrenat que hem fet servir és el que proporcionaven les mateixes llibreries OpenCV i que s'anomena *haarcascade_frontalface_default.xml*. Aquest fitxer XML està compost per diferents atributs que formen un arbre de decisió com podem comprovar a la Figura 11.

```
<_>

    <!-- root node -->
     <feature>
        <rects>
            <_>0 17 18 3 -1.</_>
             <_>0 18 18 1 3.</_>
        </rects>
         <tilted>0</tilted>
    </feature>
    <threshold>8.1820003688335419e-003</threshold>
    <left_val>-0.2865200042724609</left_val>
    <right_val>0.6789079904556274</right_val>
```

Figura 11: Part de l'arxiu *haarcascade_frontalface_default.xml* que representa un arbre de decisió.

Tot aquest procés té lloc tant a la part de l'aprenentatge com a l'hora del reconeixement i per això ha d'executar-se tant el Mac com a l'iPhone. La Figura 12 mostra com aquest mètode funciona al telèfon iPhone.

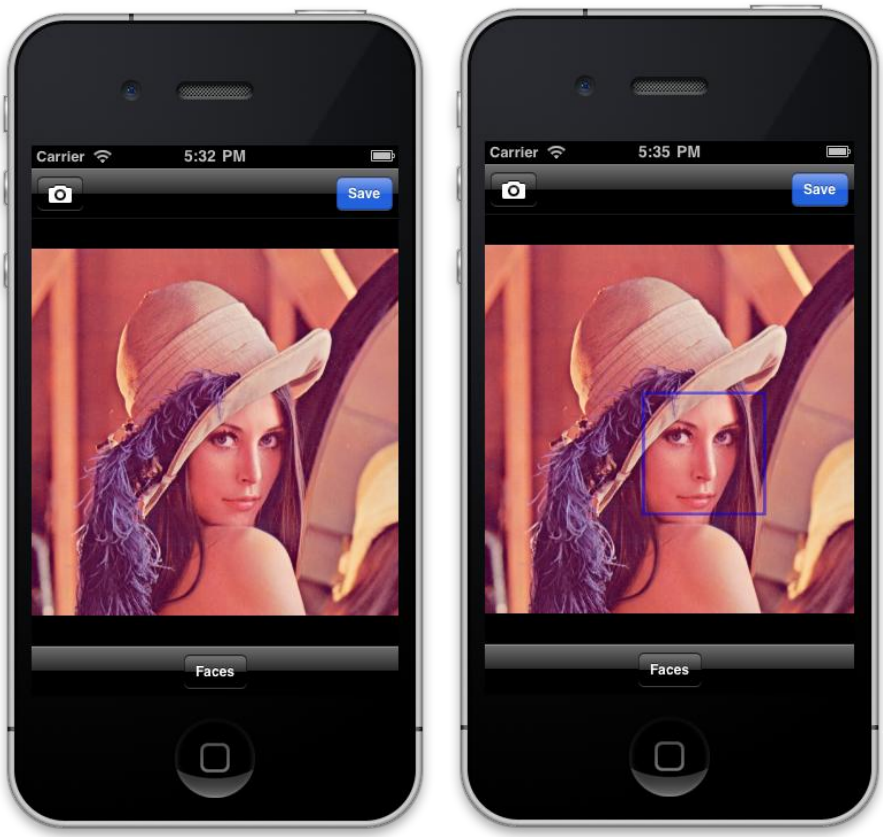


Figura 12: Procés de detecció de cares realitzat al dispositiu iPhone (simulador).

A la imatge de l'esquerra podem veure com hem carregat la popular imatge Lenna i després de pressionar el botó *Faces* veiem com, malgrat no ser una imatge frontal del tot, l'algoritme té la capacitat de retornar-nos els límits de la cara detectada els quals nosaltres després pintem amb un color blau.

5.2. PROCESSAMENT DE LES IMATGES

Un cop ja tenim la cara detectada, passem la imatge a escala de grisos i el següent pas és tractar les imatges per tal de que totes tinguin les mateixes característiques: les mateixes dimensions, la mateixa rotació de la cara, contrastos ajustats, els ulls als mateixos píxels, etc.

Per fer totes aquestes operacions hem fet servir funcions OpenCV que ens han permès executar-les tant al Mac com el dispositiu iPhone.

5.2.1. EQUALITZACIÓ DE L'HISTOGRAMA

Amb aquest procés el que aconseguim és incrementar el contrast global de les imatges, especialment quan la informació de la imatge està representada per valors de contrast molt propers. Mitjançant aquest ajustament, les intensitats estan millor distribuïdes a l'histograma [8].

L'equalització de l'histograma és útil en imatges amb fons i primers plans que són els dos clars o els dos foscos. Per exemple, s'utilitza molt per veure millor els ossos en una fotografia de raigs X o per aconseguir millors detalls en fotografies que estan sobreexposades o subexposades.

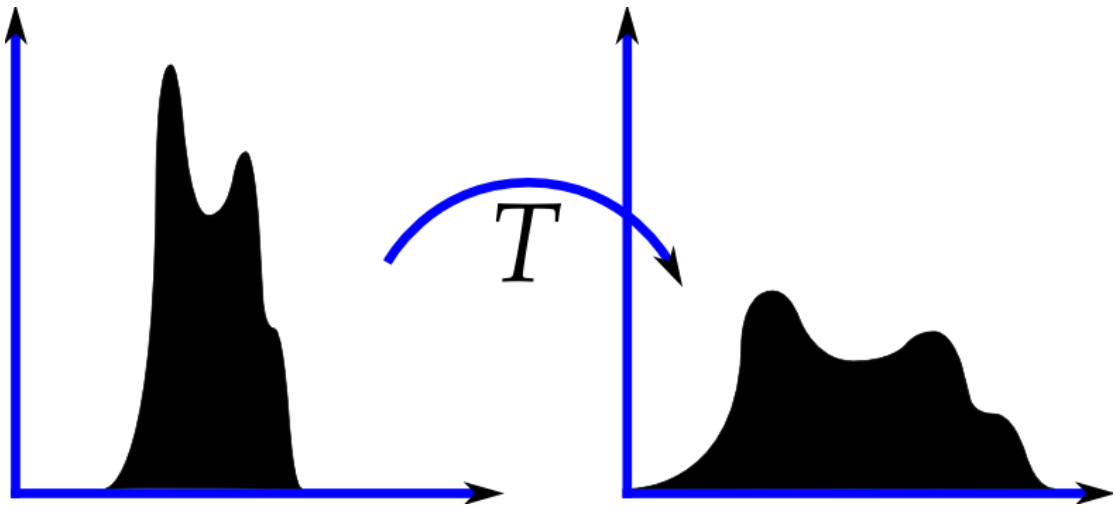


Figura 13: Efecte de l'equalització en els histogrames. Esquerra: Original; Dreta: Equalitzada.

A la Figura 13 podem observar com la imatge esquerra té totes les seves intensitats concentrades pel mig del histograma i amb un rang de valors no gaire ampli. Després de fer l'equalització podem veure que aquestes freqüències es reparteixen per totes les intensitats no tenint pics d'intensitats gaire importants i així aconseguint una imatge més uniforme.

[8] Histograma: Representació gràfica de la freqüència relativa dels nivells de gris d'una imatge

Els avantatges d'aquest algoritme són que és bastant fàcil d'implementar, que és invertible i que és computacionalment ràpid, cosa que ens interessa ja que aquesta operació també s'ha d'efectuar a l'iPhone. El desavantatge d'aquest mètode és que no té distinció, pot incrementar el contrast del soroll de fons i decrementar el senyal útil.

La funció OpenCV que realitza aquesta operació és:

- `void equalizeHist(const Mat& src, Mat& dst)`

Paràmetres: - *src*: La imatge font..
- *dst*: La imatge destí; tindrà la mateixa mida que la imatge font.

A la Figura 14 es mostra l'algoritme que s'utilitza.

1. Calcula l'histograma H de la imatge *src*.
2. Normalitza l'histograma així que la suma de tots el valors sigui 255.
3. Calcula la integral de l'histograma

   $$H'_i = \sum_{0 \le j \le i} H(j)$$

4. Transforma la imatge utilitzant H' com una taula de cerca: $dst(x,y) = H'(src(x,y))$

Figura 14: Algoritme d'equalització de l'histograma.

En la Figura 15 podem observar el resultat d'aquesta funció que apliquem abans de fer la detecció de cares per a que aquesta funcioni millor.

Figura 15: Resultat de l'equalització de l'histograma. Dreta: Imatge original; Centre: Imatge en grisos; Dreta: Imatge amb equalització de l'histograma.

5.2.2. ALINEACIÓ DELS ULLS

Per realitzar l'alineació d'ulls, primerament els hem de detectar i per fer-ho utilitzarem la mateixa tècnica dels *Haar-Like Features* que vam utilitzar per la detecció de cares i que està explicada a l'apartat 4.1. L'única diferència és que en comptes d'utilitzar un classificador en cascada per detecció de cares, utilitzem un per l'ull esquerra i un altre per l'ull dret i fem la mateixa operació però en dos passos.

Hem implementat una millora per tal de no obtenir falsos positius i ha estat reduir l'espai de cerca i per això busquem l'ull esquerra en la meitat superior dreta de la fotografia i l'ull dret a la meitat superior esquerra. El resultat de tot aquest procés és el que podem observar a la Figura 16.

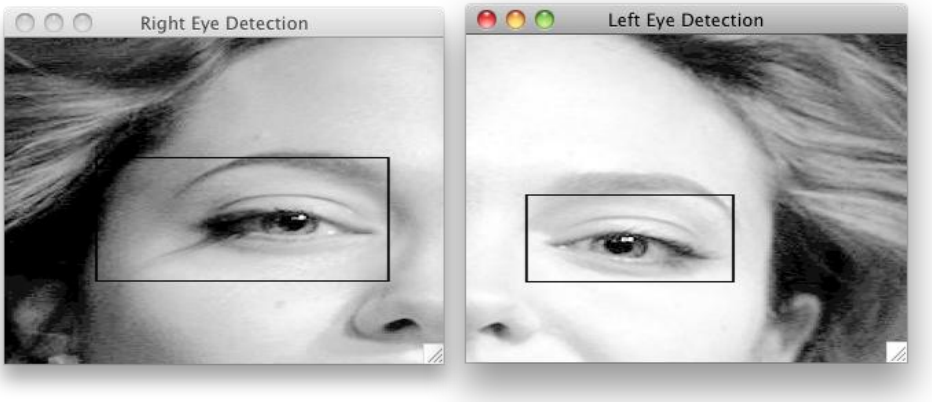


Figura 16: Resultat d'aplicar la detecció d'ulls.

També podem observar que el resultat de l'algoritme de detecció ens retorna un marc rectangular on està l'ull, per això, per calcular el centre, tracem dues diagonals des dels vèrtexs i el centre estarà en el punt de tall (Figura 17).

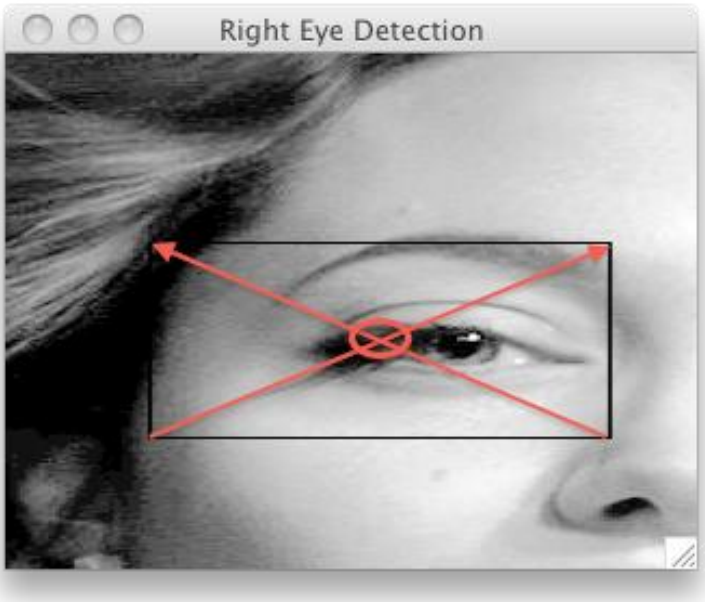


Figura 17: Centre del marc rectangular de l'ull.

Hi ha vegades que aquest centre que calculem no és el centre real i no coincideix amb l'iris de l'ull. Per aquest motiu s'ha intentat utilitzat altres tècniques sense èxit com per exemple la de trobar cercles mitjançant la transformada d'Hough [6].

Malgrat aquest petit error de trobar el centre de l'ull, el següent pas que fem és el d'alinear els dos ulls a la mateixa horitzontal. Per aquesta finalitat, simplement utilitzem alguns conceptes bàsics de trigonometria per tal de calcular l'angle de rotació (Figura 18).

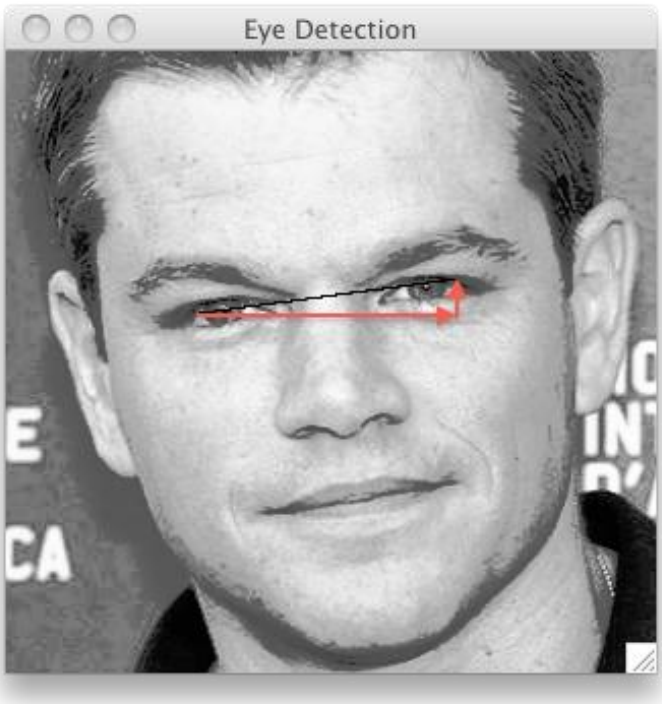


Figura 18: Imatge amb els ulls detectats però sense rotar.

L'angle de rotació, amb el seu signe corresponent, correspondran amb l'angle que hi ha entre la línia negra i la línia vermella horitzontal. Un cop trobem aquest angle, podem aplicar una de les funcions de transformació geomètrica de les llibreries OpenCV, en el nostre cas *cvGetQuadrangleSubPix.*

- `void cvGetQuadrangleSubPix (const CvArr* src, CvArr* dst, const CvMat* mapMatrix)`

Paràmetres: - *src*: Imatge font.
- *dst*: Imatge destí.
- *mapMatrix*: Matriu de transformacií 2x3 $[A \mid b]$.

El que fa aquesta funció és extreure els píxels de la imatge font i els guarda a la imatge de destí de la següent manera:

$$dst(x,y) = src(A_{11}x' + A_{12}y' + b_1\,, A_{21}x' + A_{22}y' + b_2)$$

on

$$x' = x - \frac{width(dst) - 1}{2}\,, y' = y - \frac{height(dst) - 1}{2}$$

i

$$mapMatrix = \begin{bmatrix} A_{11} & A_{12} & b_1 \\ A_{21} & A_{22} & b_2 \end{bmatrix}$$

Pel nostre fi, la variable de *mapMatrix* tindrà els següents valors:

$A_{11} = \cos(rotationAngle)$

$A_{12} = \sin(rotationAngle)$

$A_{21} = -A_{12}$

$A_{22} = A_{11}$

$b_1 = \text{width} * 0.5$

$b_2 = \text{height} * 0.5$

Amb aquests valors, aconseguim l'esperada rotació de la imatge que en el cas de necessitar píxels de fora de la imatge, utilitza el mode de replicació de píxels de la frontera per reconstruir els valors. Podem veure el resultat final i la reconstrucció de píxels a la cantonada esquerra superior de la Figura 19.

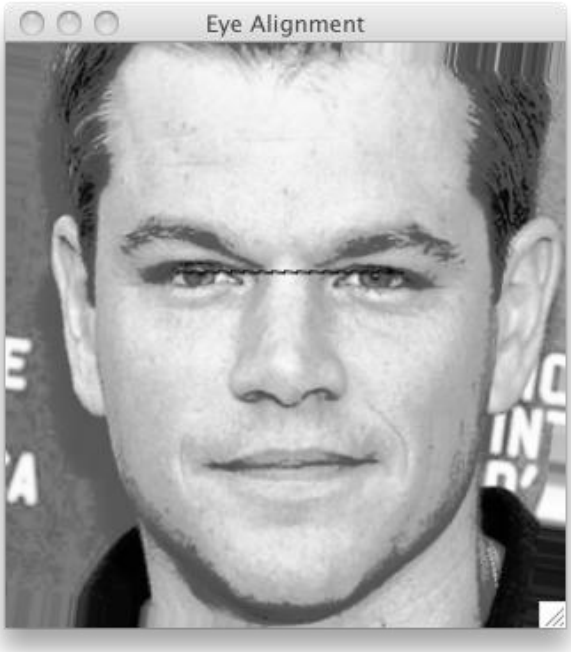


Figura 19: Resultat de la rotació.

5.2.3. ESCALAT DE LA IMATGE

El penúltim pas és l'escalat de la imatge que fem per tal que la distancia entre els ulls sigui la mateixa en totes les imatges. Aquesta distància hem decidit que sigui de 15 píxels ja que és la resolució habitual en els algoritmes de classificació facial. Per tant farem l'escalat oportú per tal d'aconseguir que la distància entre l'ull esquerre i l'ull dret sigui la desitjada.

Per aconseguir-ho hem utilitzar la funció:

- `void cvResize(const CvArr* src, CvArr* dst, int interpolation=CV_INTER_LINEAR)`

Paràmetres: - *src*: Imatge font.
- *dst*: Imatge destí.
- *interpolation*: Mètode d'interpolació:

  - CV_INTER_NN: Interpolació del veí més proper
  - CV_INTER_LINEAR: Interpolació bilineal.
  - CV_INTER_AREA: Remostreig usant la relació de les àrees dels píxels.
  - CV_INTER_CUBIC: Interpolació bicúbica.

En el nostre cas utilitzem una interpolació bicúbica ja que és la més acurada i amb la qual obtenim una imatge com la de la Figura 20.

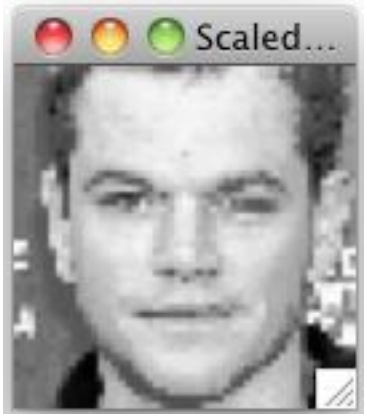


Figura 20: Imatge escalada amb distància entre ulls de 10 píxels.

5.2.4. RETALL DE LA IMATGE

Per últim, el que fem és retallar la imatge per tal de deixar els ulls de totes les fotografies en el mateix píxel i així també descartar el fons que només aporta soroll. Llavors deixarem l'ull dret del personatge al píxel (10, 10) i l'ull esquerre al píxel (25, 10). La dimensió total de la fotografia serà de 35 píxels d'ample per 33 d'alt.

Per tal de fer tota aquesta operació utilitzem la següent funció que estableix una regió d'interès (*ROI*[9]) mitjançant un rectangle proporcionat i una imatge:

- `void cvSetImageROI(IplImage* image, CvRect rect)`

Paràmetres: - *image*: Punter a la capçalera de la imatge.
- *rect*: El rectangle ROI.

Amb aquesta funció aconseguim retallar la imatge amb les nostres especificacions i el resultat és el que podem observar a la Figura 21.

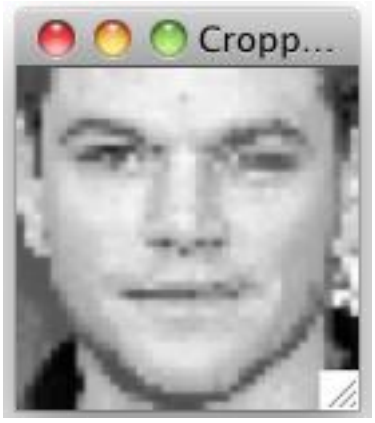


Figura 21: Imatge final un cop s'ha fet el retall del fons i s'han col·locat els ulls als píxels corresponents.

[9] ROI: Region Of Interest.

5.3. PRINCIPAL COMPONENT ANALYSIS

El Principal Component Analysis (PCA) [7] és un procediment matemàtic que utilitza una transformació ortogonal de les dades per tal de convertir un conjunt d'observacions de variables possiblement correlacionades en un conjunt de valors de variables no correlacionades anomenades components principals per tal de reduir la dimensió del conjunt inicial. Aquesta transformació està ordenada de tal forma que la primera component principal té la màxima variança de les dades possible i per tant és la que més informació conté. Aquestes variàncies també s'anomenen valors propis o *eigenvalues*[10].

El PCA construeix una transformació lineal que escull el nou sistema de coordenades pel conjunt de dades inicial en la qual la major variança és capturada en el primer eix, primera component principal. La segona variança més gran és el segon eix, segona component principal, i així successivament. Per construir aquesta transformació lineal s'ha de construir, primer, la matriu de covariança de les dades. I després cal calcular la seva descomposició en valors propis i vectors propis, també anomenats *eigenvectors*[11]. Com que la matriu és simètrica sabem que existeix aquesta descomposició dins del conjunt dels nombres reals. La transformació que passa les coordenades antigues a la nova base de coordenades és la transformació lineal necessària per tal de reduir la dimensionalitat de les dades originals.

Eigenfaces

Els *eigenfaces* són un conjunt d'*eigenvectors* utilitzats en el camp de visió per computador pel reconeixement de cares. Aquesta aproximació d'usar *eigenfaces* pel reconeixement va ser implementada per Sirovich i Kirby [8] i utilitzada per Matthew Turk i Alex Pentland [9][10] per classificació de cares. Com ja hem explicat, aquests

[10] Eigenvalues: Per cada eigenvector, el seu corresponent eigenvalue és el factor per el qual l'eigenvector canvia quan el multipliquem per la matriu.

[11] Eigenvectors: Els eigenvectors d'una matriu quadrada són els vectors, que no són igual a cero, que al ser transformats per la matriu, resten proporcionals al vector original.

eigenvectors s'aconsegueixen de la matriu de covariança. El conjunt d'*eigenfaces* pot ser calculat mitjançant el PCA explicat amb anterioritat, on simplement les matrius serien les imatges de cares humanes, i amb el següent procediment.

Primer, per crear el conjunt d'*eigenfaces*:

1. Preparar el conjunt d'imatges d'entrenament de cares. Totes les imatges que formin part del conjunt d'entrenament han de ser tractades prèviament amb el procediment comentat a l'apartat 5.2. Processament De Les Imatges. Després cada imatge serà transformada a un vector simplement concatenant cada fila de píxels en una de sola formant un vector de $r\ x\ c$ dimensions i seran guardats per columnes en una mateixa matriu $\boldsymbol{T}$.

2. Calculem la mitjana $\boldsymbol{a}$ de cada vector (imatge) i després la restem a cada un dels vectors de $\boldsymbol{T}$.

3. Calculem la matriu de covariança $\boldsymbol{S}$.

4. Calculem els *eigenvectors* i *eigenvalues* de la matriu de covariança $\boldsymbol{S}$. Cadascun dels *eigenvectors* tenen la mateixa dimensió que les imatges originals i és per això que els podem veure com una imatge i per aquesta raó són anomenats *eigenfaces*.

5. Escollim les components principals, és a dir, els D vectors propis amb major valor propi (en valor absolut).

Tot aquest procés és pot fer relativament fàcil amb les llibreries OpenCV i amb les següents funcions:

- `void cvCalcEigenObjects( int nObjects, void* input, void* output, int ioFlags, int ioBufSize, void* userData, CvTermCriteria* calcLimit, IplImage* avg, float* eigVals )`

Paràmetres: - *nObjects*: Número d'objectes d'entrada.
- *input*: Punter a la matriu d'imatges d'entrada.

- *output*: Punter a la matriu d'*eigenfaces* de sortida.
- *ioFlags: Flags* d'entrada/sortida.
- *ioBufSize:* Mida del búffer d'entrada/sortida.
- *userData:* Punter a l'estructura que conté tota la informació necessaria per les funcions de retorn.
- *calcLimit:* Criteri que determina quan parar el càlcul dels *eigenfaces*.
- *avg:* Imatge mitjana.
- *eigVals:* Punter al vector d'*eigenvalues* ordenats de forma descendent.

A les imatges Figura 22 i Figura 23 podem veure els *eigenfaces* i la imatge mitjana resultant d'aplicar la funció descrita al nostre conjunt d'entrenament.

Figura 22: Setanta primers *eigenfaces* del nostre conjunt d'entrenament.

Figura 23: Imatge mitjana del nostre conjunt d'entrenament.

A la Figura 22 els *eigenfaces* estan ordenats tal que els que tenen més informació estan els primers i els que retenen menys trets característics estan els últims si fem un recorregut d'esquerra superior a dreta inferior.

- `void cvEigenDecomposite( IplImage* obj, int nEigObjs, void* eigInput, int ioFlags, void* userData, IplImage* avg, float* coeffs )`

Paràmetres:
- *obj*: Objecte d'entrada.
- *nEigObj*: Número d'*eigenfaces*.
- *eigInput*: Punter a la matriu d'*eigenfaces*.
- *ioFlags: Flags* d'entrada/sortida.
- *userData:* Punter a l'estructura que conté tota la informació necessaria per les funcions de retorn.
- *avg:* Imatge mitjana.
- *coeffs:* Coeficients que es calculen de sortida.

Aquesta funció és la que ens projectarà cada imatge del conjunt d'entrenament i les noves imatges a classificar en l'espai de dimensions calculat amb anterioritat. En el nostre cas, mitjançant un estudi previ, hem determinat que només necessitarem un total de 220 *eigenfaces* amb els quals retenim un 98% de la informació, que és més que suficient per el nostre projecte.

Un cop hem fet aquest procediment per a totes les imatges d'entrenament podem guardar el resultat en un arxiu en format XML i així no hem de repetir aquest pas cada cop que volem classificar una nova instancia. Això ens va molt bé per exportar aquesta operació a l'iPhone ja que podem fer l'entrenament a l'ordinador i portar l'arxiu XML, que ocupa poc espai, al dispositiu mòbil i ens estalviem espai i temps de computació. Per guardar els resultats en un arxiu hem fet servir la funció:

- `void cvWrite( CvFileStorage* storage, const char* name, const void* structPtr, CvAttrList attributes=cvAttrList(),int flags=0 )`

Paràmetres: - *storage*: Arxiu d'emmagatzament.
- *name*: Nom de l'estrectura a guardar.
- *structPtr*: Punter a les dades de l'estructura a guardar.
- *attributes:* Llista d'atributs.
- *flags: Flags* d'operació.

Amb aquesta funció guardem qualsevol variable en un arxiu especificat.

Finalment, a l'hora de l'arribada d'una nova instancia a classificar, carreguem l'arxiu amb totes les dades de l'entrenament, projectem la nova instancia i ja la podem comparar i classificar amb algun algoritme de classificació, en aquest cas, el *K-Nearest Neighbor*.

Limitacions i avantatges del PCA

Algunes de les limitacions d'aquest algoritme són que quan apliquem el mètode a imatges és molt sensible als canvis d'il·luminació (les primeres components principals són normalment la il·luminació). Una altra limitació és que és una extracció de característiques no supervisada (no es tenen en compte les etiquetes).

Malgrat les limitacions o desavantatges, la seva principal avantatge és que és molt ràpid i que necessita molt poca memòria i això és perfecte per la nostra aplicació per l'iPhone.

5.4. K-NEAREST NEIGHBOR

El *K-Nearest Neighbor (K-NN)* [11] és un mètode per classificar nous objectes basat en la proximitat dels objectes d'entrenament a l'espai de característiques.

Aquest algoritme és un dels més simples de l'aprenentatge automàtic: un objecte es classifica mitjançant la majoria de vots dels seus veïns, li assignem la classe més comú entre els seus *K* veïns més propers. En el cas de que *K* sigui igual a 1, llavors simplement li assignem la classe del seu veí més proper.

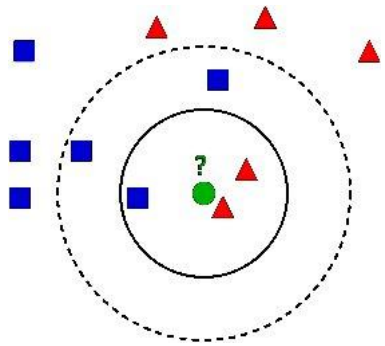


Figura 24: Diagrama exemple del K-NN.

Els veïns són agafats des d'un conjunt d'objectes dels quals coneixem la seva classificació correcta, per això necessitem un conjunt d'entrenament.

Algoritme

Els exemples d'entrenament són vectors en un espai característic multidimensional, i específicament en el nostre cas és un espai de 220 dimensions ja que vam reduir la dimensionalitat mitjançant el PCA. Llavors, els valors dels atributs del i-èssim element es representaria com:

$$cara_i = (cara_{i\ 1}, cara_{i\ 2}, \dots, cara_{i\ 220})$$

Donat un nou exemple a classificar, es calcula la seva distancia amb tots els exemples i la classe d'aquest nou element vindrà determinada pels *K* més propers.

Per a la classificació de gènere, hem utilitzat la distància euclidiana [12] mesurada amb tots els exemples.

$$distancia\ euclidiana(cara_i, cara_j) = \sqrt{\sum_{r=1}^{220} (cara_{i\,r} - cara_{j_1})^2}$$

En el cas del reconeixement de cares utilitzem la distancia Mahalanobis [13] i es diferencia de la distancia euclidiana en que té en compte les correlacions del conjunt de dades i que és invariant en escala.

$$distancia\ Mahalanobis(cara_i, cara_j) = \sqrt{\sum_{r=1}^{220} \frac{\left(cara_{i\,r} - cara_{j_1}\right)^2}{\sigma_{i\,r}^2}}$$

on σ_{i_r} és la desviació estàndard de l'element $cara_{i\,r}$ sobre el conjunt d'entrenament.

Elecció del paràmetre K

La millor elecció del paràmetre *K* depèn del tipus de dades; generalment, valors alts de *K* redueixen l'efecte del soroll en la classificació, però fa que les fronteres entre les classes siguin menys distingibles. En el nostre cas, pel reconeixement de gènere utilitzem un valor de *K* de 3 que és el que millor resultats ens ha donat. En el cas del reconeixement facial, els valors majors a 1 no tenen sentit ja que no estem buscant una classificació exacta i ja que només tenim un exemple per cada classe.

Les llibreries OpenCV ja tenen l'algoritme del *K-NN* implementat però només utilitzen la distància euclidiana per això les hem utilitzat solament per la classificació de gènere. Pel reconeixement de cares hem fet el nostre propi algoritme utilitzant la distància Mahalanobis ja que ens donava millor resultat. A continuació mostrem les funcions de les OpenCV que hem utilitzat.

- `bool cvKNearest::train(const CvMat* _train_data, const CvMat* _responses, const CvMat* _sample_idx=0, bool is_regression=false, int _max_k=32, bool _update_base=false)`

Paràmetres: - *_traint_data*: Dades del conjunt d'entrenament.
- *_responses*: Classes de cada instancia del conjunt d'entrenament.

Aquesta funció serveix per construir el model amb tot el conjunt d'entrenament i les classes de cada instància.

- `float cvKNearest::find_nearest (const CvMat* _samples, int k, CvMat* results=0, const float** neighbors=0, CvMat* neighbor_responses=0, CvMat* dist=0)`

Paràmetres: - *_samples*: Noves instancies per ser classificades.
- *k*: Número de veïns que vols tenir en compte per tal de classificar la nova instància.

El resultat d'aquesta funció és la classe a la qual pertany la nova instància classificada respecte els *K* veïns que especifiquem a l'entrada (en el nostre cas, *K*=3 és el que ens donava millor resultat).

CAPÍTOL 6: RESULTATS

Per valorar els resultats tenim el compte el percentatge de bones classificacions de gènere i el temps que triga en fer l'entrenament i en fer la classificació. No podem valorar analíticament com és fa el reconeixement de cares ja que només és a quina persona famosa s'assembla més.

A més de valorar el nostra algoritme també hem fet proves amb el software Weka [14] (*Waikato Environment for Knowledge Analysis*) que és una eina que implementa molts dels algoritmes d'aprenentatge automàtic.

6.1. DETECCIÓ DE CARES

Els resultats de la detecció de cares amb l'algoritme del *Haar-Like Features* són molt satisfactoris. Obtenim un 100% de detecció i amb un temps mitjà de 135 mil·lèsimes de segon, sempre i quan la cara estigui frontal. Quan la cara està una mica torçada, o el cap no està complet i està una mica tallat, o la persona porta ulleres (sobretot de sol), l'algoritme comença a donar problemes però no tenim cap solució per això, l'única cosa que ha de fer l'usuari és una foto el més frontal possible i sense ulleres i això s'explicarà les instruccions d'ús. Podem veure varis exemples de deteccions de cares satisfactòries i fallides a la Figura 25.

Figura 25: Exemples de deteccions satisfactòries i fallides de cares.

Podem apreciar com a la tercera imatge no s'ha detectat bé la cara ja que el cap estava tallat. Les altres dues si que tenen la cara ben detectada tot i que la segona imatge està una mica de costat.

6.2. DETECCIÓ D'ULLS

La detecció d'ulls la fem també mitjançant la tècnica del *Haar-Like Features,* però el seu resultat no és tant bo com quan detectem cares. Aquesta detecció ha de ser bastant precisa ja que amb ella, modifiquem la imatge per a que totes tinguin els ulls alineats i a la mateixa alçada. Calculem que aquesta detecció està al voltant del 70% d'encert però a vegades, i tal com mostràvem a la Figura 17, el centre de la detecció no cau en l'iris de l'ull i per això proposem tècniques alternatives a l'apartat 7.1.

6.3. CLASSIFICACIÓ DE GÈNERE

Per a la classificació de gènere utilitzàvem l'algoritme del PCA juntament amb el *K-Nearest Neighbor* i hem obtingut els resultats mostrats a continuació.

Per fer les proves hem utilitzat la base de dades d'imatges de la següent manera:

- Base de dades utilitzada per l'aprenentatge: 509 Dones i 346 homes. 855 en total.
- Base de dades utilitzada pel reconeixement: 82 Dones i 83 homes. 165 en total.

  Veure Annex 2 – Base de Dades d'Imatges d'Entrenament per més informació sobre la base de dades utilitzada.

Durant l'entrenament hem fet la reducció de dimensionalitat amb l'algoritme del PCA i hem guardat l'arxiu XML amb les dades resultats per utilitzar en la classificació i en fer tot aquest procés triga una mitja de **46.715** mil·lèsimes de segon. Aquesta operació només es fa un cop i té lloc a l'ordinador, llavors l'arxiu XML es passa a l'iPhone.

Les proves de classificació també s'han dut a terme a l'ordinador per tal de facilitar la implementació. Recordem que la classificació es fa amb l'algoritme del *K-Nearest Neighbor* on en aquest cas miràvem només un veí (K=1) i podem veure els resultats en la matriu de confusió de la Taula VII.

Entrada\Classificat com:	Dona	Home
Dona	62	20
Home	11	72

Taula VII: Resultats reconeixement de gènere amb K=1.

Amb aquests resultats podem observar que l'algoritme del ***K-Nearest Neighbor* amb K=1** té una taxa d'encert del 81.21%. De 165 mostres testejades, hi han hagut 134 bones i 31 dolentes. En fer aquestes classificacions, l'algoritme triga de mitjana 2.501 mil·lèsimes de segon, que equival a 15.15 mil·lèsimes de segon per mostra.

Aquest mateix procediment de ***K-Nearest Neighbor*** però amb **K=3** ens va donar els millors resultats (Taula VIII) i per tant és el que hem adoptat per fer el reconeixement al nostre iPhone malgrat que triga una mica més.

Entrada\Classificat com:	Dona	Home
Dona	64	18
Home	9	74

Taula VIII: Resultats reconeixement de gènere amb K=3.

Els resultats són una mica millors que amb K=1 i tenim que la taxa d'encert és del 83.64% amb 138 classificacions bones de 165 mostres i només 27 de dolentes. Com ja hem comentat, en comptes de mirar un veí, en mirem tres i llavors aquest algoritme és una mica més lent i triga una mitja 3.197 mil·lèsimes de segon per classificar les 165 mostres i en particular, 19.37 mil·lèsimes de segon per cada una d'elles, temps que trobem insignificant i per tant vam adoptar aquest model.

Classificació de Gènere amb Weka

Hem volgut fer el mateix procediment de PCA més classificació amb l'eina Weka ja que té els algoritmes ja implementats i volíem comparar els resultats.

El primer que hem fet ha estat processar les dades originals de les imatges amb un PCA i automàticament ha estimat que teníem suficient informació amb els 112 primers *eigenfaces*, en comptes dels 220 que agafàvem nosaltres per retenir el 98% de la informació.

Un cop hem processat les dades amb el PCA, fem un *K-Nearest Neighbor* amb K=1 i obtenim els següents resultats:

Entrada\Classificat com:	Dona	Home
Dona	80	2
Home	1	82

Taula IX: Resultats reconeixement de gènere amb K=1 amb Weka.

Veiem que ja amb K=1 dona molt millor resultats que la nostra implementació amb OpenCV; un 98.18% de taxa d'encert. Amb K=3 obtenim el següent:

Entrada\Classificat com:	Dona	Home
Dona	78	4
Home	2	81

Taula X: Resultats reconeixement de gènere amb K=3 amb Weka.

Amb el paràmetre K=3, veiem que, en el cas de Weka, els resultats empitjoren una mica respecte a K=1 i es queda només amb una taxa d'encert de 96.36%, però encara superior que el nostre millor resultat amb les llibreries OpenCV.

CAPÍTOL 7: TREBALL FUTUR

En aquest capítol proposem totes les millores que es podrien fer per una futura versió de l'aplicació per tal d'aconseguir per exemple una millor detecció d'ulls, un millor percentatge d'encert de la classificació de gènere i en general per un millor funcionament.

7.1. TÈCNIQUES ALTERNATIVES PER LA DETECCIÓ D'ULLS

En capítols anteriors hem vist els problemes que hem tingut per tal de fer una bona detecció d'ulls ja que si l'algoritme detectava un ull, aquest tenia bastantes possibilitats de no estar centrat i llavors empitjorava el tractament de les imatges i el posterior reconeixement. Per tal de millorar aquest procés plantegem fer una detecció d'ulls amb el *Haar-Like Features* que ja fem servir i després fer una detecció d'iris. Aquesta detecció d'iris ja l'hem intentat implementar mitjançant la transformada d'Hough però sense èxit.

Un altre mètode que podem intentar és el trobar l'iris gràcies al canvi d'intensitat dels píxels en aquesta regió. Això ho faríem agafant la regió de l'ull detectada i recorrent-la fila per fila i veure quines files són les de més intensitat i aquestes haurien de pertànyer a les de l'iris. El problema que ens podem trobar és quan les celles estiguin també incloses a la regió de detecció, llavors les seves files també tindrien una gran intensitat.

7.2. MILLORES EN LA IL·LUMINACIÓ

Ens hem adonat que la il·luminació de les fotografies juga un paper molt important a l'hora de fer el reconeixement de cares i llavors hauríem de fer algun tractament a les imatges per tal d'homogeneïtzar-les. En aquest àmbit tenim dues propostes:

- Correcció de la il·luminació mitjançant gradients relatius [15].
- Proposta depenent dels veïns adaptatiu i integrat per millora no lineal de les imatges en color (AINDANE) [16].

7.3. MILLORES PER A LA CLASSIFICACIÓ DE GÈNERE

Per la classificació de gènere el que podem implementar és un algoritme més complex que el K-*Nearest Neighbor* ja que és el més senzill de tots i per tant, en teoria, el més ineficient. Amb el software Weka hem provat altres algoritmes ja implementats com per exemple els Support Vector Machine [17] i les xarxes neurals artificials [18].

Support Vector Machine

Un cop fetes les proves amb aquesta tècnica i amb les mateixes dades d'entrenament i de testing que vam fer servir per els resultats de l'apartat 6.2. hem obtingut els següents resultats:

Entrada\Classificat com:	Dona	Home
Dona	81	1
Home	0	83

Taula XI: Resultats reconeixement de gènere amb SVM.

Podem observar com, amb aquest algoritme, i amb kernel[12] polinomial, la taxa d'encert augmenta fins el 99.39% amb només una classificació errònia de 165.

Xarxes Neurals Artificials

També hem fet aquestes proves amb el Perceptron multicapa que és un tipus de xarxa neural. Els resultats, que els podem veure a la Taula XII, són els millors que hem aconseguit amb cap altra prova amb un 100% d'efectivitat; ha classificat correctament 165 mostres d'un total de 165.

Entrada\Classificat com:	Dona	Home
Dona	82	0
Home	0	83

Taula XII: Resultats reconeixement de gènere amb Xarxes Neurals.

[12] Kernel: Funció que serveix per projectar unes dades a un espai de característiques de major dimensió per augmentar la capacitat computacional de les màquines d'aprenentatge lineal.

Aquest mètode de les xarxes neurals és el que implementaríem en una hipotètica futura versió de l'aplicació donada la seva efectivitat del 100%.

7.4. ALTERNATIVES PEL RECONEIXEMENT FACIAL

El reconeixement de cares ho fèiem utilitzant la tècnica dels EigenFaces més un *K-Nearest Neighbor*. El que podem fer per intentar millorar aquest apartat és utilitzar tècniques de reducció de la dimensionalitat més complexes com per exemple:

- FisherFaces [19][20].
- Mètodes Kernel [21][22].

No ho hem pogut implementar però segurament, al ser algoritmes més complexos i complerts, ens hauria de donar millor resultats que el PCA.

CAPÍTOL 8: CONCLUSIONS

Per concloure, pel que fa els resultats obtinguts, podem dir que hem assolit el principal objectiu d'aquest projecte en el temps establert per la planificació, que ha estat el crear una aplicació que mostri el doble d'una persona i que funcioni millor que les que hi ha al mercat.

Els algoritmes utilitzats proporcionen una alta taxa d'encert i efectivitat: més d'un 83% en la classificació de gènere i un bon detector d'ulls, que és més que suficient per a que l'aplicació sigui coherent amb els dobles que troba de les fotografies d'entrada. A més, no utilitzem internet per buscar cap doble, factor que ens avantatge sobre alguns dels nostres competidors ja que les seves aplicacions necessiten internet per processar les imatges ja que els algoritmes es troben a servidors propis.

Pel que fa al disseny, tant l'Anna com en Jordi han fet un disseny atractiu i modern de les pantalles i del logotip que ajudaria molt per a la possible venta de l'aplicació a l'App Store en un futur.

Des del punt de vista personal puc dir que he complert amb els objectius pels quals vaig escollir un projecte d'aquestes característiques:

- ✓ Aprofundir els coneixements sobre visió per computador, aprenentatge automàtic i intel·ligència artificial.
- ✓ Aplicació de els mètodes d'intel·ligència artificial i visió per computador a un problema real.
- ✓ Auto aprenentatge del llenguatge Objective-C per la programació d'aplicacions en l'iPhone.
- ✓ Perfeccionament de la utilització de les llibreries OpenCV.
- ✓ Gestió d'un projecte d'aquest abast i d'un grup de persones, en el nostre cas, dissenyadors.

Per finalitzar, reiterar que els resultats de les proves analítiques demostren que els algoritmes utilitzats són bons per els nostres objectius i que l'aplicació, amb unes 100 persones famoses a la base de dades, dóna resultats coherents. No podem assegurar que si augmentem el nombre de famosos de la base de dades, aquests resultats es mantinguin però les ampliacions proposades per versions posteriors ajudarien molt a l'eficàcia de la classificació de gènere i amb el procés de trobar el doble i aquest seria molt més acurat.

ANNEXOS

ANNEX 1 – IMATGES INTEGRALS

Una imatge integral (o també coneguda com *Summed Area Table*) és un algoritme per generar una ràpida i eficient suma de valors en una subàrea rectangular d'una quadrícula. Va ser primerament introduït al 1984 [23] però no va ser àmpliament utilitzat fins que Viola-Jones [24] ho va fer servir per el seu *framework* de detecció d'objectes.

El valor de la imatge integral a qualsevol punt (x,y) és la suma dels píxels per sobre i a l'esquerra d'aquest punt.

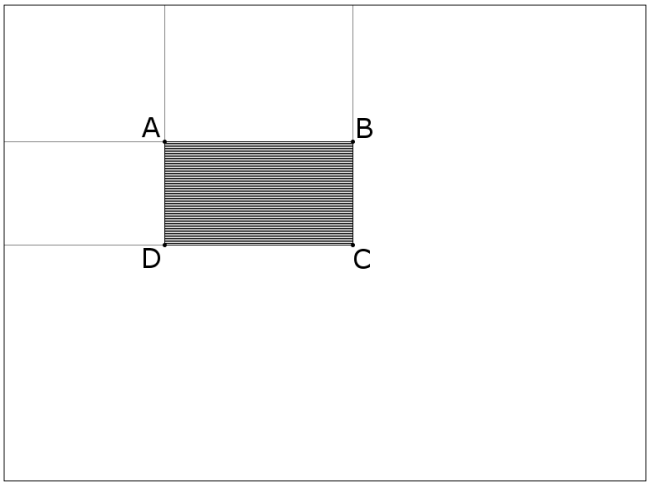


Figura 26: Exemple del càlcul de la imatge integral.

Algoritme

$$I(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x', y')$$

A més a més, la imatge integral pot ser calculada eficientment d'una sola passada sobre la imatge, utilitzant el fet de que el valor de la imatge integral al punt (x,y) és solament:

$$I(x,y) = i(x,y) + I(x-1,y) + I(x,y-1) - I(x-1,y-1)$$

Un cop la imatge integral ha estat calculada, la tasca d'avaluar qualsevol altre rectangle pot ser acomplida en temps constant amb solament quatre referencies a una matriu. Específicament, utilitzant la notació de la Figura 26, el valor solament és:

$$\sum_{\substack{A(x) \le x' \le C(x) \\ A(y) \le y' \le C(y)}} i(x',y') = I(A) + I(C) - I(B) - I(D)$$

ANNEX 2 – BASE DE DADES D'IMATGES D'ENTRENAMENT

Les imatges que s'han utilitzat per l'entrenament han estat obtingudes per Aleix Martínez i Robert Benavente [25] al Centre de Visió per Computador (CVC). Conté més de 4.000 imatges en color que corresponen a 126 persones diferents. Les imatges són frontals i amb diferents expressions facials, condicions d'il·luminació i d'oclusió (ulleres de sol i bufandes).

Les imatges van ser fetes al CVC sota condicions controlades. No es van imposar restriccions en la roba dels participants, ni en el maquillatge, ni en el cabell, etc.

ANNEX 3 – BASE DE DADES D'IMATGES DE FAMOSOS

Per la base de dades d'imatges de famosos contem amb unes aproximadament 100 imatges de celebritats. Són imatges de famosos d'arreu del món i pràcticament totes són frontals. Alguns exemples de persones famoses que tenim a la nostra base de dades són l'actor americà Jonnhy Depp, o l'actiu Angelina Jolie o el futbolista anglès David Beckham.

REFERÈNCIES BIBLIOGRÀFIQUES

[1] ES Academic. http://www.esacademic.com/dic.nsf/eswiki/287535.

[2] University of Leeds, http://www.comp.leeds.ac.uk/vision/opencv/opencvref_cv.html.

[3] Viola, Paul i Jones, Michael; “Rapid Object Detection using a Boosted Cascade of Simple Features”, on Computer Vision and Pattern Recognition, 2001.

[4] Lienhart, R. i Maydt, J., "An extended set of Haar-like features for rapid object detection", 2002.

[5] Marc Fernandez Girones, “Classificació de Cares Segons la Seva Postura”, Projecte Final de Carrera de l’Enginyeria Tècnica en Informàtica de Sistemes, 2008.

[6] Richard O. Duda i Peter E. Hart, “Use of Hough transformation to detect lines and curves in pictures”, 1971.

[7] Pearson, K; "On Lines and Planes of Closest Fit to Systems of Points in Space" Philosophical Magazine 2, 1901.

[8] L. Sirovich and M. Kirby, “Low-dimensional procedure for the characterization of human faces”, Journal of Optical Society of America, vol. 4, 1987.

[9] M. Turk i A. Pentland, "Face recognition using eigenfaces". Proc. IEEE Conference on Computer Vision and Pattern Recognition, 1991.

[10] Turk i A. Pentland, "Eigenfaces for recognition". Journal of Cognitive Neuroscience 3, 1991.

[11] Belur V. Dasarathy, Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques, 1991

[12] Wikipedia. http://en.wikipedia.org/wiki/Euclidean_distance.

[13] Mahalanobis, P C, "On the generalised distance in statistics". Proceedings of the National Institute of Sciences of India 2, 1936.

[14] Weka. http://weka.wikispaces.com/

[15] Zujun Hou i Wei-Yun Yau, “Relative gradients for imatge lighting correction”, Institute for Infocomm Research, A-Star, Singapore, 2010.

[16] Li Tao i Vijayan Asari, “An integrated neighborhood dependent approach for nonlinear enhancement of color images”, ITCC, 2004.

[17] Corinna Cortes i V. Vapnik, "Support-Vector Networks", Machine Learning, 20, 1995. http://www.springerlink.com/content/k238jx04hm87j80g/

[18] Rosenblatt, F., "The Perceptron: A Probalistic Model For Information Storage And Organization In The Brain”, 1958

[19] K. Etemad, R. Chellappa, “Discriminant Analysis for Recognition of Human Face Images”, Journal of the Optical Society of America A, Vol. 14, 1724-1733.

[20] W. Zhao, R. Chellappa, A. Krishnaswamy, “Discriminant Analysis of Principal Components for Face Recognition”, Proc. of the 3rd IEEE International Conference on Face and Gesture Recognition, FG'98, 1998.

[21] F.R. Bach, M.I. Jordan, “Kernel Independent Component Analysis”, Journal of Machine Learning Research, Vol. 3, 2002.

[22] M.-H. Yang, “Face Recognition Using Kernel Methods, Advances in Neural Information Processing Systems”, T. Diederich, S. Becker, Z. Ghahramani, Eds., 2002.

[23] Crow, Franklin; “Summed-area tables for texture mapping”, SIGGRAPH, 1984.

[24] Viola, Paul i Jones, Michael, “Robust Real-time Object Detection”, International Journal of Computer Vision, 2002.

[25] Aleix Martínez, http://www2.ece.ohio-state.edu/~aleix/ARdatabase.html.

Agraïments:

Primerament vull agraïr a l'Àgata Lapedriza i al Xavier Baró per tot el suport i ajuda altruista que m'han donat. Per descomptat, al meu director de projecte Jordi Gonzàlez que s'ha portat molt bé i ha valorat el meu treball i esforç.

També volia donar les gràcies a la meva novia Jade Catalano que m'ha estat ajudant en tot el possible i d'ella va ser la idea d'aquesta aplicació i a la família.

Finalment, mencionar a en Jordi Puig i l'Anna Sanz que han col·laborat en aquest projecte fent el disseny gràfic.

Gràcies!

Signat: .....

Marc Fernández Gironès

RESUM

En aquest projecte es presenta l'aplicació per a dispositius mòbils *Doppelganger*. La seva funció és, a partir d'una fotografia, detectar la cara i mostrar la persona famosa de la nostra base de dades que més s'assembla a la persona en la fotografia.

Per la implementació s'han utilitzat algoritmes de visió per computador i d'aprenentatge automàtic com per exemple el PCA i el *K-Nearest Neighbor*, tot utilitzant llibreries gratuïtes com són les *OpenCV.*

RESUMEN

En este proyecto se presenta la aplicación para dispositivos móviles *Doppelganger.* Su función es, a partir de una fotografía, detectar la cara y mostrar el personaje famoso que más se parece a la persona de la fotografía.

Para la implementación se han utilizado algoritmos de visión por computador y de aprendizaje automático como por ejemplo el PCA y el *K-Nearest Neighbor*, utilizando librerías gratuitas como las *OpenCV.*

ABSTRACT

In this project we present the mobile devices application called *Doppelganger*. Its goal is to detect the face from a picture and to show the celebrity from our database that is most similar to the person in the photo.

Computer vision and machine learning algorithms, from the Open Source libraries *OpenCV,* such as the PCA and the *K-Nearest Neighbor* have been used for the implementation.