Aplicació mòbil que utilitza l'algoritme ASM per resoldre la constància de color

Jordi Garcia Altimiras

Resum—El cervell humà és capaç de mantenir el color dels objectes encara que variï la il·luminació. Aquest procés s'anomena constància de color (Colour constancy). En aquest treball s'ha agafat un algoritme que intenta emular el cervell humà per resoldre la constància de color de manera senzilla i automàtica, anomenat "Adoptive Surround Modulation" o "ASM" i s'ha adaptat, prèviament en Matlab, a Android amb l'ajuda de la llibreria OpenCV perquè es pogui utilitzar en dispositius mòbils com ara un telèfon mòbil o un "tablet", sempre hi quan tingui Android de sistema operatiu.

Paraules clau—Constància de color, Matlab, Android, Adapative Surround Modulation, ASM, il·luminació, dispositiu mòbil,aplicació, OpenCV

Abstract—The human brain is capable of maintining the colour of objects even if the illumination changes. This procces is called Colour constancy. In this project we choose and algorithm, named Adapative Surround Modulation or ASM, that tries to emulate the human brain to solve colour constancy easly and automatic and we adapted it to Android, previously in Matlab, with hellp of OpenCV library so it could be used in portable devices such as mobile devices or tablets if they have Android as operating System.

Index Terms—Colour constancy, Matlab, Andoird, Adaptative Surround Modulation, illumination, mobile devices, application, OpenCV

1 Introducció

EL color és una part molt important en la vida de les persones, tant a nivell visual com emocional. El color ens permet distingir els diferents objectes del fons. Per nosaltres, el color dels objectes no varia durant en tot el día malgrat la gran quantitat de canvis d'il·luminació que podem experimentar. Aquesta habilitat que té el nostre cervell s'anomena constància de color, "Colour constancy", [6].

Actualment no se sap del cert quines àrees del cervell en són responsables, tot i això, molts grups d'investigació les agrupa segons el nivell neuronal en el que actuen:

- Nivell sensorial
- Nivell perceptual
- Nivell cognitiu

La contribució relativa d'aquests nivells encara esta en debat. Malgrat això, molts investigadors afirmen que aquests fenomen pot ser explicat per mecanismes de baix nivell presents en la retina i en les àrres V1 i V4 del còrtex visual.

Aquest problema s'ha intentat resoldre de dos maneres diferents, intentant simular els mecanismes de baix nivell o basats en aprenentatge previ d'una base de dades.

Adaptative Surround Modulation, ASM, correspon al grup que intenta simular el cervell humà. Concretament intenta simular les neurones V1 i V4, ja que es creu que colour constancy és més un problema de normalització i no d'aprenentatge.

Com es pot comprovar en l'atricle [5]. ASM ha estat comparat amb altres algoritmes en diferents conjunts d'imatges, i s'ha comprovat que dóna millors resultats que els d'aprenantatge. Per tant, tenim un algoritme genèric i automàtic amb millors resultats que un algoritme que ha aprés en el conjunt corresponent.

Colour constancy és un terme desconegut per a gran majoria de gent i crec que és molt interessant que arribi a més gent, sobretot persones que es dediquen a la fotografia perquè els pot ajudar a entendre millor com funciona la il·luminació i com funcionen les càmeres que utilitzen. A més, tindrén una nova eina, amb ASM, per retocar les fotografies.

Aquesta aplicació doncs permetrà a la gent, no només poder obtenir una imatge on hi ha eliminada la il·luminació si no que a més, podran aplicar-hi la il·luminació que ells vulguin, per així, poder comprovar com canvia el color de les coses depenent de la llum que hi ha en l'escena.

Tot seguit, explicaré tot el procés de desenvolupament

- *E-mail de contacte: jordi.galt@gmail.com*
- *Menció realitzada: Enginyeria del Software*
- *Treball tutoritzat per: C. Alejandro Párrega (Ciències de la Computació)*
- *Curs 2016/17*

d'aquesta aplicació, explicant com s'ha adaptat el codi de Matlab a Android, com s'ha organitzat l'aplicació perquè sigui senzilla, els problemes trobats i els resultat obtinguts.

2 Objectius

L'objectius principal d'aquest treball és, com s'ha comentat anterioriment, el de desenvilpuar una aplicació per a Android que adapati ASM.

A part de l'objectiu anterior també hi ha els següents objectius:

- L'aplicació ha de ser senzilla i intuïtiva.
- Ha de tenir l'opció de poder utilitzar la càmera, si el dispositiu en disposa.
- Ha de poder agafar imatges del dispositiu.
- Ha de permetre a l'usuari guardar les fotos al seu dispositiu.
- Ha de permetre a l'usuari canviar la il·luminació de les fotos.

Aquests objectius serveixen per tenir una guia del que serà el treball i poder fer un calendari amb cada tasca.

3 Adapative Surround Modulation

Quan es realitza una forografía ens quedem amb una imatge RGB, on cada píxel té un valor per el color vermell, un per el color verd i un per el blau. El conjunt d'aquests tres valors, que van de 0 a 255 per cada color, és el color resultant que veiem en la imatge.

El problema sorgeix quan hi ha una gran diferència entre el color més clar que capta la càmera i el més fosc. Això se soluciona de varies maneres. Una és adaptar aquesta diferència entre 0 i 255, resultant amb imatges molt fosques degut a que el píxel mes clar tindrà un valor de 255 però la resta tindran un valor de 10.

Un altre manera és truncar els píxels més clars a 255 i deixar la resta igual. Aquesta tècnica dóna com a resultat una imatge no tant fosca però hem perdut informació de la il·luminació que tenia la imatge.

Adaptative Surround Modulation intenta solucionar aquest problema detectant les zones més clars. En aquestses hi distingeix dues parts, el centre i el voltant on obtenim dues gaussianes. La interacció entre el centre i el voltant es resol utilitzant una diferència de gaussianes, DoG. Per què la DoG ens pugui fer una bona estimació de la il·luminació s'ha de trobar l'amplada del kernel de la gaussiana correcte i el pes òptim de la funció gaussiana.

Per evitar que els paràmetres anteriors siguin fixes, per tant que variïn en funció de la imatge, ASM adapta tres propietats de les neurones corticals (àrea V1):

- La mida mínima del camp receptiu varia en relació al contrast local dels estúmuls.
- La influència de la part que rodeja al centre varia depenen del contrast local tant del centre com del voltant, amb major inhibició per contrasts alts en els estúmils.
- El camp receptiu cortical augmenta el seu diàmetre sistemàticament per aproximadament un factor de 3 desde àrees petites a grans.

El resultat és una model de la imatge on es poden distingi les àrees més clares. A partir d'aquí, s'agafa aquest model i s'implementa una simulació de l'àrea 4 per trobar la il·luminació de la imatge.

S'agrupen els píxels de la imatge resultant a partir del seu valor d'il·luminació en un histograma. Apartir d'aquí en quederem amb els valors més grans, el tant per cent que agafarem és calculat a partir del model resultant fent que sigui diferent per cada imatge. D'entre aquests valors ens quederem amb el valor més petit.

Un cop tenim la il·luminació per cada canal, aquesta és multiplica a la imatge original. La imatge llavors es torna a escalar perquè quedi entre 0 i 255 i tenim la imatge final sense la il·luminació estimada.

A l'apèndix hi ha un esquema de com funciona l'algorisme. Per una explicació molt més detallada del funcionament de l'algorisme llegiu l'altricle [5].

4 Implementació en Android

En aquest apartat explicaré com he implementat l'algorisme en Android. El codi original està en Matlab i el podreu trobar al següent enllaç goo.gl/nQUenN.

4.2 Codi en Android

Per implementar aquest codi en Android, per tant en Java, he utilitzat una llibreria de tractament d'imatge, OpenCv.

He decidit utilitzar aquesta llibreria perquè Matlab, a diferència de Java, està optimitzat per treballar amb matrius. Aquesta lliberia és una llibreria especialitzada en tractament d'imatges i conté gairebé totes les funcions que necessito per poder implementar el codi correctament.

Aquesta llibreria conté un objecte de matriu, Mat, que em permet convertir un Bitmap (imatge RGB) en un matriu amb tres canals, on cada canal és un color, he hagut de tenir en compte que en matlab les imatges es guarden en RGB sent el primer canal vemell, segon verd i tercer blau, en openCV les imatges guardades en un Mat té els canals al revés BGR.

A més, aquesta llibreria hi ha implementades la majoria de funcions pròpies de Matlab que tracten amb matrius, tot i que, algunes s'han hagut d'implementar desde zero.

Per fer-ho de manera ordenada i per no perdre'm jo en el codi, vaig decidir dos coses abans de començar.Primer miraría de que l'estructura fos al màxim d'igual possible a l'estructura del codi en Matlab i que les variables i funcions es diguessin igual. D'aquesta manera facilita la depuració del codi i mirar els resultats ja que, en principi, hi ha el mateix tant en un com l'altre.

Una altra cosa important ha sigut la simplificació del codi. En Matlab el codi esta preparat per que es puguin indtroduïr paràmetres, a part de la imatge, i si no s'introdueixen paràmetres hi ha uns per defecte. Com que l'aplicació està pensada per usuaris que no han de saber com funciona ASM no tenen perquè saber quins paràmetres d'entrada han d'introduïr, per tant, la implementació en Android no té la opció de poder introduir-los. A més, hi ha càlculs al llarg del codi que es poden simplificar al no introduir paràmetres degut a que el resultat serà sempre al mateix.

4.2.1 Funcions OpenCV

OpenCv m'ha ajudat enormement amb les funcions i objectes que conté. Comentaré les funcions que més he utilitzat.

Filter2D: La funció més important que he hagut d'utilitzar. Agafar una matriu i la convergeix amb el kernel passat. A més té una sèrie d'opcions per configurar-la. S'utilitza gariabé 50.

Threshold i compare: Dues funcions que comparen matrius, matriu amb matriu o matriu amb un valor. El resultat és una matriu de 0 i 1/255 on 0 vol dir que no és compleix la comparació.

També les dos funcions *Split i merge* que divideixen una matrius de tres canals, BGR, en tres matrius de un canal per poder treballar cada canal per separat.

Per últim comentar la funció *calcHist* on retorna l'histograma en forma de matriu.

A part de les funcions comentades també he utilitzat totes les funciones relacionades en sumar, restar, dividir, multiplicar matrius. Per més informació en les funciones d'OpenCv i OpenCv en general, referència [3].

4.2.2 Funcions Implementades

Tot i que openCv conté la majoria de funcions necessàries he hagut d'implementar un parell de funcions pròpies de matlab.

Stdfilt: Calcula la desviació estàndard de una imatge.

Linspace: Retorna un vector amb valors linealment espaiats.

imQuantize: Quantitza la imatge utilitzant específics nivells de quantització i valors de retorn.

Aquestes funcions, tot i no ser molt difícils d'implementar no contava amb el temps que em portaría. Això ha comportat a que hem retresés amb el temps establert inicialment per adapatar l'algoritme. Per veure com estàn implementades en Matlab referència [2].

5 DESENVOLUPAMENT DE L'APLICACIÓ

En aquest apartat explicaré el procés seguit per desenvolupar l'aplicació. Així com els problemes trobats i la solucions aplicades.

5.1 Metodologia

La metodologia aplicada en el desenvolupament del treball es pot dividir en diferents etapes o processos, que es troben directament relacionats amb les fases del projecte.

Podem determinar les següents etapes segons la metodologia aplicada:

1- *Etapa d'Anàlisi:* En aquesta primera fase o etapa es porta a terme un anàlisi del projecte. S'estudia el problema plantejat i es defineixen les diferents tasques a realitzar diferenciant entre les mes costoses i les que menys.
Després es realitzar una planificació temporal. En aquesta s'estima el temps que portarà cada tasca i l'ordre en que es realitzaran estudiant quines tasques son mes prioritàries i quines menys.. Així es té un ordre a seguir durant el transcurs del treball, a més, es podrá veure si el treball va endarrerit i buscar-hi solucions a més d' estudiar perquè s'ha enderreit.
En aquesta fase també es fa un estudi de les tecnologies actuals que poguin ser semblant o ens poguin ajudar a realitzar el treball.
També es realitza un primer disseny de l'aplicació en paper, per poder ensenyar als stakeholders i perquè ens ho apovin.

2- *Etapa de Desenvolupament:* L'etapa més llarga sense cap dubte. En aquesta es desenvolupa l'aplicació guiant-nos de tota la planificació realitzada en l'etapa anterior. S'utilitzar una metodologia RAD, [1], ja que es demana l'opinió dels stakeholders contiunuament per trobar qualsevol error en

l'aplicació el més ràpid possible tant si és d'implementació o problema del disseny inical.

3- *Etapa de testeig:* Fase realitzada en paral·lel a l'anterior.
Com s'ha comentat anteriorment l'aplicació esta en constant revisió dels stakeholders. A més però, en la part més costosa de l'aplicació, l'adaptació del codi de Matlab a Android, hi ha una depuració constant. S'ha comprovat que els resultats de l'aplicació eran els mateixos que els del codi constanment. Això té conseqüències, desenvolupem un codi de millor qualitat però el desenvolupament és més lent, però. ens ha permés saber en cada moment com evoluciona l'aplicació.

Per poder realitzar de manera controlada i organitzada el projecte s'ha utilitzat l'entron de desenvolupament Android Studio (IDE),[4], i GitHub (repositori Git), [8], com a control de versions i repositori. D'altra banda tota la documentació realitzada s'ha guardat en Dropbox, [10], perquè pogui ser accessible desde qualsevol lloc i dispositiu.

5.2 Disseny de l'aplicació

Les aplicacions Android s'organitzan en Activitys, on cada Activity és una pantalla visible per a l'usuari. Després d'analitzar l'aplicació i del feedback dels stakeholders l'aplicació consta de les següents activitys o vistes:

1. Pantalla inicial: Pantalla que s'obre només en obrir l'aplicació. Consta de 3 botons.
    a. Càmera: Obra la càmera del dipositiu per poder tirar una foto.
    b. Galeria: Obra la galeria del dispositiu per seleccionar una foto.
    c. Llista d'il·luminacions: Canvia a l'Activity llista d'il·luminacions.

2. Llista d'il·luminacions: Activity on tenim un llistat de totes les il·luminacions que tenim guardades. Podem seleccionar o afegir.
3. Detall il·luminacó: Activity on es pot modificar, afegir o eliminar una il·luminació.
4. Detall Imatge: Activity on es realitza la part més important de l'aplicació, conté diferentes opcions:
    a. Processar imatge: Permet aplicar ASM a la imatge.
    b. Seleccionar Il·luminació: Permet seleccionar una il·luminció de la llista.
    c. Afegir il·luminació: Permt aplicar la il·luminació seleccionada anteriorment.
    d. Guardar imatge: Permet guardar la imatge que es mostra per pantalla al dispositiu.
    e. Obtenir il·luminació: Fa la mitja de cada canal de la imatge per calcular la il·luminació, [7].(Grey World)

El diagrama que hi ha a continuació mostra com es relacionen les diferent Activitys entre elles. Totes les activitys poden tornar enrere amb el botó del dispositiu. La Galeria i la Càmra no són Activitys pròpiaes de l'aplicació, sinó que obra les pròpies dels dispositiu. Les fletxes amb punts vol dir que esperan un resultat de l'Activity on apunten, per tant un cop s'ha aconseguit el resultat es torna a l'activity anterior.

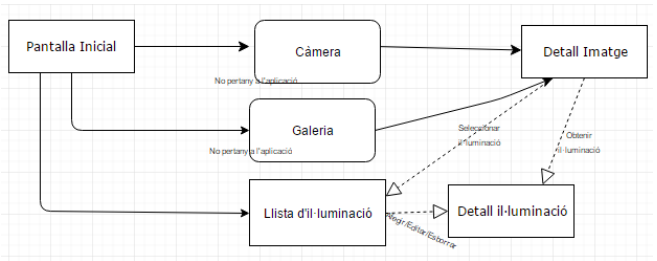


Figura 1: Relació entre les Activitys

5.3 Estructura del codi

Una aplicació android té tres carpetes molt diferenciades i molt important les tres per el correcte funcionamet de l'aplicació.

La primera es diu "manifests". En aquesta hi ha un arxiu xml anomenat "AdroidManifest.xml". En aquest manifest es declara els permisos que necessita l'aplicació, en aquest cas l'ús de la càmera i el permis per escriure a l'emmagatzament extern. També s'indica les diferents activitys que hi ha a l'aplicació, sinó estàn posades aquí no es podran utilitzar.

La segona carpeta es diu "java". En aquesta hi ha tota la part lògica del programa, així com tot el codi en java de l'aplicació. Aquí podem distingir dos tipus de fitxer java. El primer són tots els arxius java que fan referència a una activity, cada activity té el seu arxiu java associat on hi ha la lògica d'aquestsa. El segon són els arxius java que no estàn relacionats en cap activity. Aquests arxius són classes de java. Cal destecar-ne tres dintre d'aquesta aplicació "Process Image", "Shared Pref" i "Luminance". La primera és la classe on hi ha l'adaptació de l'algoritme ASM i Grey World. En la segona hi ha la lògica per afegir, editar i guardar les il·luminacions i en la tercera hi ha la classe objecte d'il·luminació.

La tercera, i última carpeta es diu "res". Aquí hi ha tots els recursos que utilitzarà l'aplicació. Aquesta carpeta té un seguit de subcarpetes:

a. *Layout:* conté els xml de les activitys. En aquests xml es defineix el disseny que tindrà cada activity.
b. *Menú:* L'aplicació conté un menú en dos de les activitys. El disseny d'aquests es troba en

aquesta carpeta en format xml.

c. *Values:* L'aplicació suporta tres idiomes; català, castellà i anglès. Depenent en quin idioma tinguis el dispositiu l'aplicació estarà en un o altre. Les traduccions de cada paraula de l'aplicació es troben en aquesta carpeta.

Android Studio com hem pogut veure té tots els fitxers molt ben organitzats. Existeix un altre carpeta "jniLibs" on hi ha tot el codi de la lliberia OpenCv. Aquesta carpeta però, no hi he de treballar.

Per veure el codi en Java i com està implementada tota l'aplicació, [9].

6 FUNCIONAMENT DE L'APLICACIÓ

He explicat com he implementat l'algoritme ASM, el disseny de l'aplicació i com està estructurat el codi. En aquest apartat explicaré com funciona tota l'aplicació.

6.1 Gestió de les imatges

L'aplicació desenvolupada la seva principal funció és treballar amb imatges, per això s'han de gestionar correctament.

La imatge que es processarà a l'aplicació es pot obtenir de dues maneres, a través de la càmera o a través de la galeria. Tant la galeria com la càmera no són pròpies de l'aplicació, sinó que quan es selecciona un o l'altre s'obre l'aplicació del dispositiu.

Un cop tenim la fotografia aquesta s'envia a l'activity "DetailPicture". En aquesta activity es mostra la imatge per pantalla i pots seleccionar si li vols afegir una il·luminació o li vols treure l'actual.

El problema aquí es que si la imatge és molt gran no es pot mostrar per pantalla, això passarà sempre amb les imatges fetes des de la càmera del mòbil. Per això, abans de poder visualitzar-la s'ha de comprovar si té el tamany correcte o s'ha d'escalar perquè pogui visualitzar-se.

Un altre problema és que les imatges no tenen perquè estar guardades amb la rotació correcte. Això fa que les imatges es poguin veure girades. Per solucionar això, he utilitzat "ExifInterface". Aquesta interfície conté informació de la posició en que estaba el dispositiu, sempre i quan el dispositiu tingui giroscopi, poden així calcular quina rotació se li ha d'aplicar a la imatge perquè es vegi correctament.

Finalment, la imatge es pot guardar des de l'activity poden així guardar les tranformacions que se li hagin aplicat.

6.2 Gestió de les il·luminacions

Un altra part important en l'aplicació és gestionar les il·luminacions per poder aplicar-les a la imatge.

Una il·luminació és una clase java, Luminance. Aquesta clase conté el valor vermell, verd i blau corresponent a la il·luminació, el nom i una id generada aleatoriament.

Per poder guardar-les i que no s'eliminin un cop es tenqui l'aplicació s'ha utilizat "Shared Preferences". Shared Prefernces és un interfície que proporciona Android per guardar informació de l'aplicació. Utilitza el sistema clau-valor.

El problema aquí és que no volem guardar un valor sinó un conjunt d'objectes. Per solucionar això guardem totes les il·luminacions en una llista, aquesta llista la convertim en un Json utilitzant la llibreria Gson, [11]. El que guardem a SharedPreference és el json resultant de la llista de totes les il·luminacions.

Sempre hem de tenir a l'aplicació el Json actualitzat, per això cada modificació que es faci a una il·luminació, sigui crear, editar o esborrar es tornarar a fer el Json de la llista d'il·luminacions.

Al diagrama que hi ha a continuació es mostra com funciona aquest procés.

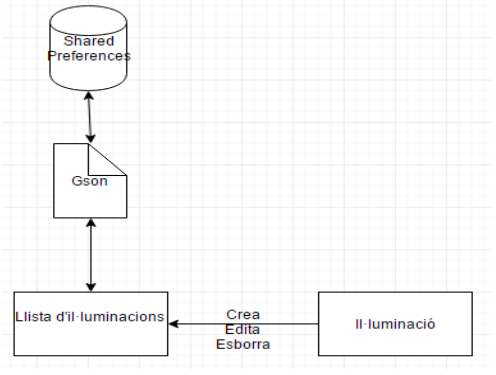


Figura 2: Procés de guardar i agafar les il·luminacons de l'estructura Shared Preferences.

Com podem observar a l'activity Detall il·luminació i mostrem l'objecte il·luminació. Un cop finalitzada aquesta acció, aquest objecte s'afageix a la llista i aquesta llista es converteix en un Json per poder-ho guardar a Shared Preference. Si volem recuperar la llista hem de fer el procés a l'inversa.

Un cop tenim la llista si volem modificar o eliminar una il·luminació primer l'hem d'identificar. Això ho fem a partir de la id propia de cada il·luminació.

6.3 Gestió dels algoritmes ASM i Grey-World

En l'aplicació s'utilitzen dos algoritmes. ASM per treure la il·luminació de una imatge i Grey-World per identificar la il·luminació de la imatge.

Aquests dos algoritmes están dins la mateixa classe de Java, ProccesImage. En aquesta classe hi han les funcions necessàries per poder-los aplicar a la imatge.

Els dos algoritmes estan implementats amb l'ajuda d'openCv.

En aquesta classe però només es realitzan les modificacions al Mat corresponen. La feina de convertir-lo en bitmap i mostrar-lo per pantalla es realitza a l'activity "Detall Imatge".

6.4 Gestió dels idiomes

L'aplicació suporta tres idiomes, català, castellà i anglès. L'idioma que li apareixerà a l'usurai serà al mateix en que hi tingui el dispositiu. Si l'idioma del seu dispositiu no és cap dels anteriors li apareixerà l'anglès per defecte.

Android facilita que les aplicacions puguin donar suport a més d'un idioma habilitant la carpeta Strings. En aquesta carpeta hi ha el llistat de totes les strings que hi ha a l'aplicació i les seves traduccions.

A més, té una pestanya exclusiva en que de manera fàcil pots afegir idiomes i escriure la traducció de cada string guardada.

6.5 Gestió del permisos

L'aplicació necessita que se l'hi dongui dos permissos. Poder utilitzar la càmera del dispositiu i poder escriure a l'emmagatzematge intern.

En Android inferior a la versió 6.0 els permissos es donaven quan l'aplicació es baixava de Google Play. A partir d'Android 6.0 hi ha dos tipus de permisos. Els permisos "no perillosos" on totes les aplicacions els tenen i els permisos "perillosos". Aquests permisos se'ls hi demanarà a l'usuari mentre utilitza l'aplicació si els hi vol donar o no.

Com que hi ha moltes versions d'Android i molta gent ja té la 6.0 he implementat la gestió del permissos de la versió 6.0.

Quan un usuari, amb versió 6.0 o superior, entri a l'aplicació li sortiran un avís en que necessita donar permís a l'aplicació per utilitzar la càmera i poder escriure a l'emmagatzematge.

Si l'usuari no dona aquests permisos no podrà utilitzar la càmera i no podrà guardar fotos des de l'aplicació.

6 RESULTATS

En aquesta secció mostraré els resultats obtingut amb l'aplicació així com el temps d'execució i ho compararé amb els resultat obtinguts utiitzant el codi en Matlab.

Els resultats no seran exactament iguals. Això és degut a dos factors, quan descomprimim la imatge en matlab i en Android obtenim un resultat diferent. Aquest resultat no es percep amb la vista però si mirem els valors dels píxels i els comparem veurem que són lleugerament diferents.

Un altre factor és que si la imatge és molt gran no podrem treballar amb ella en Android si abans no la fem més petita. Això comporta pèrdua d'informació però si no la reduïm el telèfon mòbil no té suficient memòria per processar-la.

Aclarat aquests punt procedim a mostrar els resultats. En aquesta taula hem comparat 3 imatges les quals són iguals però amb il·luminació diferent. Les imatges les trobareu a l'anexxe.

Hi ha tres resultats, resultat amb Android, resultat amb Matlab i resultat amb Matlab amb la imatge reduïda igual que Android. Els valors tenen el color del seu canal per fer-ho més llegible.

	Android		**Matlab(sense reducció)**		**Matlab(amb reducció)**	
Imatge 1	52,65 87,86	41,89	109,34 231,59	101,25	117,33 240,56	111,85
Imatge 2	52,29 34,58	32,05	167,07 67,24	56,06	183,78 91,98	75,52

Taula 1: Comparació Android vs Matlab

Com podem observar els valors són molt diferents, sobretot entre la versió Android i les dos de Matlab. Tot i això, veiem que entre ells tenen una relació semblant. El canal amb el valor més alt és el mateix que en els tres casos, igual que el més petit.

A continuació hi ha les imatges resultants dels tres algoritmes i la imatge original.

	Imatge1	Imatge 2
Android		

Matlab (sr)		
Matlab (ar)		
Original		

Taula 2: Comparació visual Android vs Matlab
Sr = sense reducció, ar = amb reducció

Com podem observar a les imatges, les imatges no són excessivament diferents entre si. Potser la diferència més visible a simple vista és en la imatge 2 on la versió en Android té la llum vermella més remarcada que en les version en Matlab. També podem veure com en la Imatge 1 tant la versió d'Android com la de Matlab a canviat el color de la llum de blau a verd i ha eliminat molta de la brillantor que feia la llum blava.

Les següents imatges que analitzarem no han sigut reduïdes ja que el seu tamany era suficientment petit per el dispositiu mòbil. La taula segueix la mateix estructura que la taula 1.

	Android		**Matlab**	
Imatge 3	106,96 199,72	120,73	103,29 214,66	126,17
Imatge 4	194,56 23,34	217,64	185,10 24,79	217,64

Taula 3: Comparació Android vs Matlab v2

Com podem observar, aquest cop els resultats son molt més semblants comparats amb els de la Taula 1. Això és degut a que no hem tingut de fer una reducció per tant, no s'ha perdut informació.

A continuació les imatges resultants de passar la imatge per Android o per Matlab, originals a l'annexe.

	Imatge 3	Imatge 4
Android		
Matlab		
Original		

Taula 4: Comparció visual Android vs Matlab v2

Com podem observar, a diferència de les altres imatges, en aquestes la diferència és mínima. El més destecat és que la imatge 4 en matlab és una miqueta més clara que la versió en Android. Per la resta la diferència no es distingeix a través de l'ull humà.

Mirant les dos taules podem veure com l'algoritme funciona gairebé amb els mateixos resultats tant amb imatges molt grans com en imatges més petites.

Les següents proves però, confirmen que per molt que per l'ull humà les diferències són mínimes, les imatges entre si són bastant diferents.

A la taula que hi ha a continuació comparem la diferència entre es il·luminacions. No hem comparat les imatges sense reducció ja que no té sentit comparar dues imatges que no tenen el mateix tamany. Les imatges de la Taula 2 tenien un tamany original de 2340x4160 i han sigut reduïdes a 1460x821. Les imatges de la Taula 4 tenen una mida de 163x222. S'ha utilitzat la fórmula "Reproduction Angular Error,[12] ,per calcular la diferència.

	σ
Imatge 1	5.07º
Imatge 2	9.78º
Imatge 3	2.56º
Imatge 4	2.58º

Taula 5: Diferència entre la il·luminació de les imatges

Com podem comprovar la diferència no és més gran

de 10º. Veiem que la imatge que dóna més diferència és la 2, que si ens fixem amb les imatges s'hi veu una clara diferència visual. La imatge 1 també es pot distingir diferència visual, tot i que més subtil. En la imatge 3 i 4 l'error és més petit que de 3º i no veiem diferències a simple vista. Podem dir doncs que una diferència menor de 3º no veirem diferència visual.

Podriem afirmar que el valor de la imatge 2 és massa elevat i hi podem veure diferències visualment. Això és degut al que s'ha comentat al principi, que Android i Matlab no llegeixen les imatges amb els mateixos valors, a més les dos primeres fotos estàn realitzades amb una càmera que li ha aplicat balanç de blanc amb algun algoritme que intenta resoldre constància de color diferemnt a ASM. Aquest fet pot ser la causa de que donguin valor més elevats ja que se li ha aplicat un algortime de correcció dos cops, primer per la càmera i posteriorment amb ASM.

La última comparació que analitzaré serà la del temps d'execució. Compararé el temps d'execució de les 4 imatges mostrades anteriorment i una imatge 16x16 que he utilitzat per comprovar que el codi funciona correctament. Totes les imatges les trobareu a l'annex.

	Temps (ms)	Tamany
Imatge 1	27723	1460x821
Imatge 2	24770	1460x821
Imatge 3	177	163x222
Imatge 4	189	162x222
Imatge prova	56	16x16

Taula 6: Temps d'execució de les imatges amb el seu tamany

Com podem comprovar com més gran és la imatge més temps està executant-se. Tot i això els temps no son excessius. Si mirem les imatges més grans, 1 i 2, el temps d'execució ronda els 27 segons. Crec que és un bon temps si tenim en compte tots els càlculs que realitza i, que si no reduïm les imatges el dispositiu no té suficient memòria. Per tant, crec que són uns temps molt bons i que he realitzat una bona feina d'optimització.

7 CONCLUSIONS

Desrpés de treballar durant un semestre en aquest treball puc dir que estic satisfet amb la feina feta i que he assolit els objectius plantejats al principi.

He aconseguit implementar satisfactòriament l'algoritme ASM a Android donant uns resultats més que acceptables tot i que no iguals al codi en Matlab per les raons comentades anteriorment.

També he desenvolupat una aplicació per mòbil totalment funcional. Aquesta aplicació apart d'implementar ASM també permer guardar il·luminacions, pot canviar la il·luminació de la imatge si es desistja i permet guardar les imatges resultants al dispositiu mòbil.

Comentar també que l'adaptació del codi m'ha portat molt més temps del que tenia previst, fent que s'enderreris la meva planificació per l'úlitm més i que no pogués acabar de polir la resta de l'aplicació.

7.1 Proposta de millores

Una millora podria ser aconseguir que el dispositiu no es quedés sense memòria si la imatge és molt gran, com també buscar la manera perquè els valors RGB de la imatge siguessin els mateixos que en Matlab.

Una altra milloria sería acabar de polir l'aplicació, això vol dir revisar que no quedin petits "bugs" en l'aplicació i que es comporta correctament i també canviar el disseny de la interfície. Actualment té el disseny per defecte i sempre és més reconeixible una aplicació si té el seu propi disseny.

També és podria fer que l'aplicació adapti més d'un alforitme apart de ASM. D'aquesta manera es podrien comprovar les diferències més fàcilment i donaria més opcions a l'usuari.

Cerc també que s'hauria de posar una "activity" que expliqui què és la constanàcia de color i perquè s'utilitza l'aplicació, ja que és un termer força desconegut entre la gent i si no se'ls hi explica anteriorment no sabran el perquè de l'aplicació.

Per últim, actualment l'aplicació per trobar la il·luminació de una imatge, quan no es fa ASM, ho fa amb l'algoritme Grey-world. Aquest algortime per algunes imatges funciona molt bé i per algunes no tant per això estaria bé que també utilitzes ASM per estimar la il·luminació de una imatge. També és podria fer que quan es fa ASM a una imatge, apart d'eliminar la il·luminació d'aquesta també es guardés la il·luminació trobada al dispositiu.

AGRAÏMENTS

Primer de tot voldria agreïr a l'Arash, autor del codi de ASM en Matlab per quedar amb mi sempre que li he demenat i resoldre'm tots els dubtes que he tingut. A més, s'ha esforçat perquè pugui entendre l'algoritme al més bé possible.

Agraïr també al meu tutor, Alejandro Parraga, per donar-me suport i ajudar-me sempre que li he demanat, sobretot en entendre com funciona ASM.

A tots els meus amics i companys que els hi he demanat una vegada i un altre que provessin l'aplicació perquè i trobessin errors.

Per últim agrair a la meva família tot el suport donat durant tots aquests mesos.

BIBLIOGRAFIA

[1] Metodologia RAD [Online]. Disponible: http://metodologiarad.weebly.com

[2] Pàgina principal Matlab [Online] Disponible: https://es.mathworks.com/

[3] Pàgina principal OpenCv [Online] Disponible: http://opencv.org/

[4] Pàgina principal desenvolupament Android [Online] Disponible: https://developer.android.com/index.html

[5] C. Alejandro Parraga i Arash Akbarinia, "Colour Constancy Beyond the Classical Receptive Field". TPAMI (under review) 2017.

[6] Colour Constancy [Online] Disponible: http://colorconstancy.com/

[7] Colour Balancing: Grey World [Online] Disponible: https://web.stanford.edu/~sujason/ColorBalancing/grayworld.html

[8] Repositori Github [Online] Disponible: https://github.com

[9] Codi Android, repositori Github [Online] Disponible: https://github.com/Guerci/AppOpencv

[10] Emmagatzematge al núvol, Dropbox [Online] Disponible: https://www.dropbox.com/home

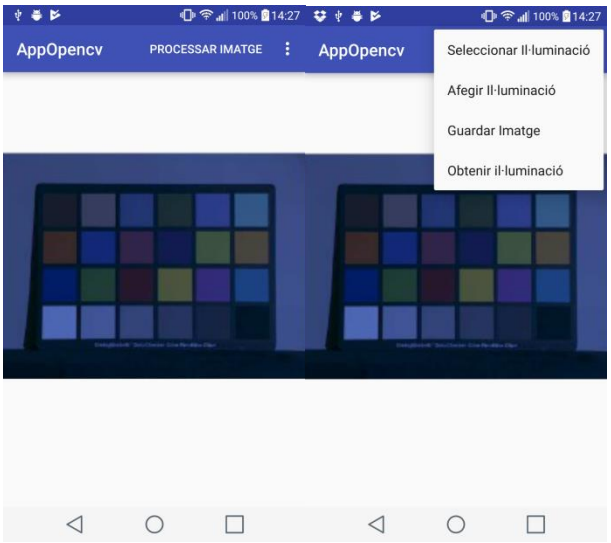
[11] Llibreria Gson [Online] Disponible: https://github.com/google/gson

[12] G. D. Finlayson and R. Zakizadeh, "Reproduction angular error:An improved performance metric for illuminant estimation," perception,vol. 310, no. 1, pp. 1–26, 2014.

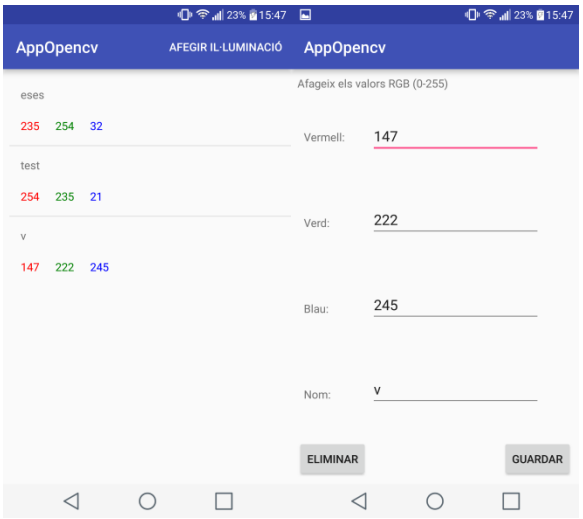
ANNEX

A1. IMATGES DE LES ACTIVITYS DE L'APLICACIÓ

Imatges del menú principal



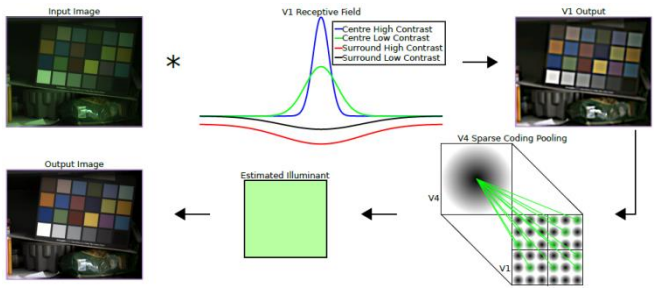
Imatges de l'activity detall imatges, una amb el menú desplegable.



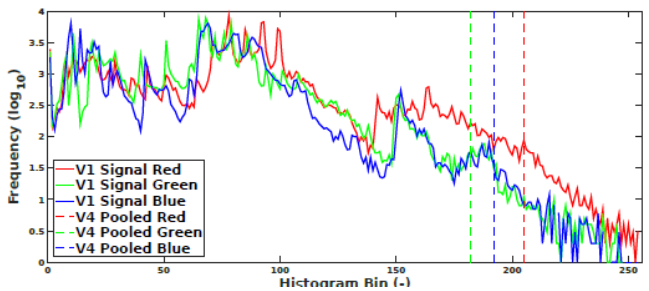
Imatges de les avtivitys llista d'il·luminacions i detall il·luminació.

**No hi ha la càmera ni la galeria ja que no pertanyen a l'aplicació.*

A2. ESQUEMA ASM



Esquema del funcionament de l'algoritme ASM.



Histograma dels valors en cada canal on es mostra per on es fa el pooling.

A3. IMATGE 16X16

Imatge 16x16 utilitzada durant el procés d'adaptació del codi.