UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Politècnica Superior de Gandia

Disseny i implementació d'una App prototip per a la generació de moodboards empleant Matlab

Treball Fi de Grau

Grau en Enginyeria de Sistemes de Telecomunicació, So i Imatge

AUTOR/A: Bañuls Granell, Neus

Tutor/a: Flores Asenjo, Santiago José

CURS ACADÈMIC: 2023/2024

RESUM

En la fase d'ideació de tot projecte gràfic, sovint és necessari plasmar les idees que estan en la ment del dissenyador o dissenyadora. Això es fa a través d'un *moodboard* o tauler d'inspiració, un conjunt d'imatges, colors, paraules i textures que serveix per a comunicar visualment una idea o un concepte.

En aquest projecte s'implementarà en MATLAB una App capaç d'integrar les eines necessàries per a automatitzar la creació d'un *moodboard*. Concretament, s'utilitzaran tècniques de correlació en 2D per a la identificació de tipografies, mètodes de segmentació per clústers per a l'anàlisi de colors dominants, i eines de *deep learning* pròpies de MATLAB per a la detecció de persones. Tot això s'integrarà en una única App que podrà servir com a prototip per a una aplicació o *plug-in* que puga ser integrada en programes de disseny com Adobe Illustrator o Adobe Photoshop.

ABSTRACT

In the ideation phase of any graphic project, it is often necessary to embody the ideas that are in the mind of the designer. This is done through a moodboard or inspiration board, a set of images, colors, words and textures that serves to visually communicate an idea or a concept.

In this project, an app which is capable of integrating the necessary tools to automate the creation of a moodboard will be implemented in MATLAB. Specifically, 2D correlation techniques will be used for font identification, cluster segmentation methods for key color analysis, and MATLAB's own deep learning tools for human detection. All this will be integrated into a single App that can serve as a prototype for an application or plug-in that can be integrated into design programs such as Adobe Illustrator or Adobe Photoshop.

PARAULES CLAU

Aplicació, *moodboard*, correlació 2D, *deep learning*, *clustering*.

KEY WORDS

Application, moodboard, 2D correlation, deep learning, clustering.

ÍNDEX

1. INTRODUCCIÓ

1.1. Justificació del tema

En el present treball, com bé s'indica al resum, la meta és la creació d'un generador de *moodboards* amb MATLAB, el qual ha de ser capaç de crear una mena de *collages* a partir de la facilitació de diverses dades per part de l'usuari.

La motivació d'aquest tema naix, en primer lloc, de la relació amb el món artístic a causa d'estudiar el Doble Grau en Enginyeria de Sistemes de Telecomunicació, So i Imatge + Comunicació Audiovisual de l'Escola Politècnica Superior de Gandia (EPSG). L'interés que han despertat el disseny i la direcció artística cursats en la part de Comunicació Audiovisual, ha suposat l'afany de voler aprofitar la creació d'un projecte tècnic per a relacionar-lo amb un propòsit artístic, per a fer d'aquest un procés multidisciplinari, però sempre centrant-se en la part que té a veure amb les telecomunicacions.

En segon lloc, el fet d'haver cursat l'assignatura "Tractament digital d'imatge i vídeo" i descobrir les possibilitats que ofereix aquest camp a l'hora de crear aplicacions útils per a la vida quotidiana, junt amb el fet d'haver cursat aquesta assignatura al quint curs, i, per tant, tenir tots els conceptes de l'assignatura recents, ha conduït a voler basar l'assaig en les distintes tècniques de correlació, segmentació i altres, vistes a classe.

A més a més, la realització del projecte final de la mateixa assignatura també ha conduït a voler aprofitar un prototip de generador de paletes de color, creat conjuntament amb l'alumne Álvaro Herrera Ramos, per a integrar-lo com a part del resultat final del generador de *moodboards*.

En quart i últim lloc, cal esmentar que, amb la recent aparició de la intel·ligència artificial (IA), s'han automatitzat moltes de les tasques humanes que solen fer-se repetitives en el dia a dia; açò ha dut a la idea de voler automatitzar algun procés o necessitat dintre del camp del disseny. Per tant, aquest és un motiu clau per a l'elaboració d'aquest treball final de grau (en avant, TFG).

Tot plegat, es pretén justificar l'ús de l'aplicació que es crearà com a *plug-in* base o com a una ferramenta continguda en aplicacions d'ordinador usades per a disseny, com ara Adobe Illustrator o Adobe Photoshop.

1.2. Objectius

Amb la realització d'aquest TFG es té com a objectiu principal la creació d'una App que genere automàticament *moodboards* a partir d'un conjunt de recursos donats per l'usuari que la mateixa aplicació li demana.

Els objectius específics que conformen el principal ja esmentat, s'exposen a continuació:

- Estudiar les tècniques de TDI per a relacionar-les amb els elements d'un moodboard.

- Testejar les possibilitats de detecció i reconeixement de tipografies i de persones amb MATLAB LiveScript.

- Esbossar la interfície de l'aplicació que es vol crear.

- Implementar el codi de l'App amb MATLAB.

- Integrar en l'aplicació un generador de paletes de color creat durant el curs acadèmic.

1.3. Metodologia

En aquest subapartat, s'expliquen tots els passos a seguir per a l'elaboració del TFG, des de la seua ideació fins al testatge de l'aplicació que es vol crear, i s'exposa la metodologia seguida en la present memòria acadèmica per a la presentació escrita del treball final de grau.

En primera instància, i com a part del procés d'ideació del TFG, després de presentar-li la idea inicial al tutor, es va haver d'aprofundir en els conceptes teòrics necessaris per a comprendre el funcionament de les tècniques de tractament digital d'imatge (en avant, TDI), refrescant així els coneixements adquirits al curs 2023-2024 i seleccionant les possibles eines per a elaborar el projecte.

Seguidament, es feren diverses proves amb MATLAB LiveScript per a veure si era factible crear l'App; així, es va concloure que, mitjançant algunes tècniques de TDI, es podria aconseguir un resultat òptim, ja que era possible detectar tipografies, persones i colors, la qual cosa era indispensable per a la realització del treball. A partir d'aquest punt, es va donar per suposat que gran part de la resta de funcions de l'aplicació es podrien realitzar, per tant, el repte era cohesionar diverses funcions en un mateix *plug-in* que funcionara de manera raonable. Finalitzada aquesta etapa, es va polir i es va presentar la proposta final de TFG.

Ara bé, pel que fa a la metodologia seguida a la memòria escrita, inicialment es proposa assentar les bases teòriques del treball a tall d'introducció, on es deixe ben clar què és un *moodboard*, què és el tractament digital de la imatge i què és MATLAB.

A continuació, la idea és endinsar-se en la teoria vista a classe per a posteriorment justificar el plantejament de l'aplicació, de manera que s'explica com funciona la correlació, la segmentació, els descriptors i el *deep learning*, incloent-ne exemples de cadascuna de les tècniques. La principal font d'informació en aquest punt són els apunts de classe proporcionats pel professor.

Successivament, ja es pot manifestar el plantejament de l'aplicació generadora de *moodboards*; com que ja es té clar què és un *moodboard* i quines són les tècniques de TDI que interessen per a la creació del projecte, es poden relacionar aquests dos punts a manera d'explicació de les distintes funcions que integra l'App. Per consegüent, es presenten totes les proves realitzades amb MATLAB durant el procés d'ideació esmentat abans, per a l'enteniment de les funcions o ferramentes del *plug-in*, incloent-ne l'explicació del generador de paletes de color ja creat com a projecte final de l'assignatura. Com a resultat, es du a terme l'esbós del generador de *moodboards.*

En la darrera etapa es tracta el marc pràctic, on s'implementa, al cap i a la fi, l'aplicació amb MATLAB AppDesigner, es presenten els resultats obtinguts i, per acabar, es declaren unes conclusions finals sobre el TFG.

2. BASES TEÒRIQUES DEL TFG

2.1. El *moodboard* en el context del disseny

Els creatius i les creatives que treballen en algun dels àmbits de l'art, necessiten una manera d'agrupar elements d'inspiració i referències que se solen recollir en la fase inicial a la creació d'un projecte. Actualment, la ferramenta del *moodboard* o taula d'inspiració permet encarnar eixes idees de forma concreta i visual.

Un *moodboard* es pot elaborar en format físic o digital, recollint imatges, textos, colors o materials en un mateix espai de creació, com si es tractara d'un collage. La disposició d'aquests elements no sol importar, les imatges apareixen muntades unes damunt d'altres, hi ha objectes retallats per la seua silueta, poden aparéixer algunes paraules arreu de tota la taula (això sí, sense tapar les imatges)... No obstant això, el seu grau d'elaboració es pot dur fins al nivell que es desitge; al fi i a la cap, la idea és reunir el nombre més gran d'idees en una mateixa àrea.

Elaborar el *moodboard* en físic és un procés costós, ja que s'ha de buscar inspiració en suports gràfics com revistes, llibres, fotografies, i altres, la qual cosa suposa invertir molt de temps en el procés de documentació: però, si s'opta pel seu format digital, aquest procés de cerca es redueix a minuts, perquè existeixen pàgines d'inspiració en línia com Pinterest, Tumblr, Behance o Instagram, els quals disposen d'un potent cercador web d'imatges. Així i tot, queda donar-li forma al tauler per a obtenir el resultat final, i aquest sol ser un procés una mica tediós. Ací està el quid de la qüestió, que es comentarà en el punt 2.2.

Amb aquest collage ja finalitzat, dissenyadores gràfiques, de moda, d'espais, directores d'art, de cinema, pintores, escultores i, fins i tot, cuineres, poden orientar als seus clients o a elles mateixes d'acord amb l'estètica i el camí que seguirà el seu projecte. Malgrat totes les variants del camp artístic, en aquest TFG es treballarà entorn del *moodboard* per al disseny gràfic, la moda i la direcció d'art (que està dins de la branca del cinema), acotant així els elements a imatges, siluetes de persones, colors i textos. A continuació, es mostren alguns exemples de *moodboards* de les branques esmentades.

Figura 1. *Des de l'esquerra, moodboards de disseny gràfic, de moda i de direcció d'art.*

2.2. Introducció al tractament digital de la imatge

El TDI és una mena d'estudi sobre els processos mecànics i algorítmics que permeten, o bé millorar la informació visual d'imatges per a la interpretació humana, o bé processar les seues dades per a la percepció autònoma per part de màquines; per al primer cas, s'empren processos senzills, com el realçament, la restauració i la compressió d'imatges, mentre que per al segon, s'usen tècniques més complexes, com ara el reconeixement de rostres i el processament d'empremtes dactilars.

La seua història es remunta a la dècada del 1940 amb el desenvolupament de les computadores modernes, que ja incloïen memòria programable i transistor. Amb els anys, el seu ús es va fer més i més rellevant, fins que als anys setanta ja s'utilitzava aquesta matèria per a facilitar alguns processos en la medicina, la geografia i l'astronomia. En l'actualitat, el continu avanç tecnològic i l'expansió de la Internet han impulsat el creixement i les aplicacions del processament digital d'imatges.

De l'ampli ventall d'aplicacions que abasta, totes elles es desenvolupen a través de programes en ordinadors, ja que les imatges no són més que matrius de píxels. Com ara, la ferramenta MATLAB és una de les millors opcions per a treballar de forma bàsica amb imatges, perquè disposa de moltes llibreries i funcions que permeten resoldre els problemes de TDI ràpidament. Processar una imatge es pot traduir en el fet que uns algoritmes modifiquen una imatge digital per a crear-ne una nova, extraient o potenciant allò que es vol d'ella.

Es pot dir que el TDI està format per dues parts, les quals es corresponen amb els seus dos objectius (la interpretació humana i la percepció autònoma d'imatges). Primerament, està la més bàsica i la que s'ha de tenir més clara, que és on es tracten tècniques de realç de punt, de filtratge, de restauració, transformacions geomètriques i més (tècniques que van des de l'eliminació del soroll d'una imatge fins a la detecció de característiques en ella). La segona part és més complexa, es basa en la visió artificial, on es troben la segmentació, la morfologia i els descriptors d'imatges, la detecció de vores i els mètodes de classificació i de detecció (que es poden utilitzar, per exemple, en la robòtica industrial per a la lectura de codis, el recompte d'objectes o el control de qualitat, entre altres).

Igualment, depenent del processament que es faça a la imatge, es poden involucrar diversos nivells de complexitat: nivell baix, nivell mitjà i nivell alt. El nivell baix inclou operacions bàsiques com l'eliminació de soroll i la millora de contrast, on les entrades i les eixides són imatges. El nivell mitjà se centra en segmentació, la detecció de vores i la descripció d'objectes; les entrades solen ser imatges, però les eixides són atributs, com vores o contorns. El nivell alt involucra tasques més avançades, com la visió per ordinador, l'anàlisi de formes i funcions cognitives associades amb la visió, que sovint requereixen la presa de decisions (igual que la intel·ligència artificial) i aprenentatge automàtic.

Ací, sempre es parla d'un procés lineal pel qual es va passant per a obtenir una classificació d'imatges i detecció d'objectes concrets, que són aquells que es volen visualitzar. El procés consta de cinc fases: l'adquisició, la segmentació, la descripció, la detecció i el reconeixement d'objectes, seguint sempre aquest mateix ordre; una etapa rere d'altra. En la **Figura 2**, s'il·lustra esquemàticament aquest procés, el qual, a continuació, s'explica més en detall.

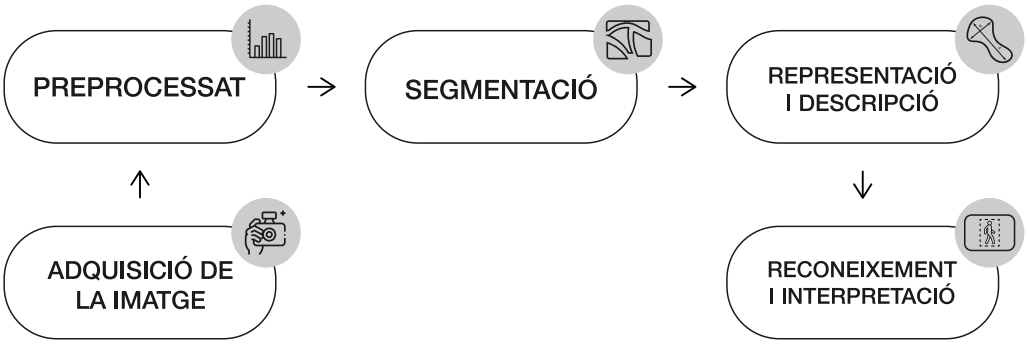


***Figura 2**. Fases del processament d'imatges.*

Sempre es comença amb l'adquisició, que és la manera amb la qual s'obtenen les imatges. En aquesta part, s'estudien conceptes d'òptica i de fotografia. Convé conéixer aquest procés per a identificar fàcilment si cal corregir alguna característica de les imatges.

Després, ve la part del preprocessat, una etapa fonamental on es realitzen els ajustos que s'acaben de mencionar, és a dir, que es prepara la imatge per a la seua correcció mitjançant una sèrie de tècniques, com la reducció de soroll, de distorsió o la uniformització de la il·luminació, abans de sotmetre-la al processat. Amb açò, l'objectiu és millorar la qualitat de la imatge i simplificar la posterior anàlisi.

En tercer lloc, està la fase de segmentació. Generalment, en el TDI es parla d'un projecte de visió artificial, per tant, cal separar i indicar quines parts o regions té una imatge i quines són d'interés, centrant-se en els objectes concrets que es vulguen visualitzar.

La quarta etapa consta de la representació i descripció, que es tradueix en mesurar els objectes que ja s'han separat per a, posteriorment, prendre decisions de què és cadascun d'ells segons la seua grandària, elongació, orientació i altres paràmetres.

En últim lloc, a la part de reconeixement i interpretació, s'identifica o es classifica informació significativa dintre d'una imatge, és a dir, que ja es pot identificar què és cada objecte mitjançant processos computacionals, com si es tractara d'una ment humana; en aquest punt, les tasques de decisió, d'aprenentatge profund i de *computer vision*, estan relacionades amb el funcionament de l'actual intel·ligència artificial.

Canviant de tema, el camp del tractament digital de la imatge es nodreix de moltes matèries vistes al llarg del grau de les telecomunicacions, sent recomanable, però no indispensable, tenir coneixements en el tractament digital del senyal (d'on s'extrau la major part d'informació útil), del tractament digital d'àudio, els sistemes de TV i vídeo (pel que fa als formats de les imatges), de flux de dades multimèdia i, sobretot, d'instal·lacions audiovisuals (al fi i a la cap, es treballa amb imatge i vídeo). En definitiva, aquesta és una disciplina molt relacionada amb molts aspectes de l'enginyeria.

També, es pot destacar que un dels aspectes més interessants del TDI és que, amb tenir coneixements teòrics i pràctics d'aquesta matèria, es poden entendre i implementar les tècniques del processament digital d'imatges, arribant al punt d'obtindre

totes les capacitats necessàries per a planificar i implementar un sistema complet de TDI, com ara, aplicacions realistes de visió artificial.

Si bé és cert, el camp de les telecomunicacions sols treballa el processament digital de la imatge des del punt de vista de l'enginyer, desenvolupant la mateixa ferramenta o els *plug-ins* necessaris per a acomplir una tasca que, normalment, pertany al món del disseny, com és l'edició d'imatges amb programes com Adobe Photoshop o Adobe Lightroom.

En resumits comptes, el processament digital d'imatges és una ampla disciplina amb diverses aplicacions, des de la millora de la qualitat visual fins a la percepció autònoma per part de màquines. En aquest TFG, s'implementaran els aspectes del TDI oportuns per a la identificació i el reconeixement d'elements, que es complementaran amb altres funcions pertinents mitjançant MATLAB.

2.3. El software MATLAB

Avui dia, els ordinadors són punts clau en molts àmbits laborals, permetent càlculs ràpids i precisos. MATLAB destaca en matemàtiques i estadística, sent preferit per a càlculs, algoritmes i simulacions.

En concret, MATLAB és fonamental en el tractament digital d'imatges. Per exemple, la *Image Processing Toolbox*[1] ofereix una àmplia gamma de funcions, conegudes com a *M-files* o *M-functions*, per a abordar diverses fases del processament d'imatges.

MATLAB inclou funcions per a preprocessament, com carregar imatges i ajustar contrast i soroll, fonamentals per a la integritat de les dades. També, proporciona filtres per a suavitzar, realçar o eliminar soroll, millorant la qualitat visual i destacant característiques clau per a l'anàlisi. Per a la segmentació, ofereix algoritmes per a dividir imatges en regions significatives o identificar objectes d'interés basats en color o textura. En l'extracció de característiques, MATLAB identifica vores, vèrtexs i punts d'interés, clau en l'anàlisi i reconeixement de patrons. A més a més, facilita la reconstrucció d'imatges i l'anàlisi estadística, que és essencial per a comprendre i comunicar resultats.

Amb una integració completa de càlculs, visualització i programació en un entorn amigable, aquest *software* és ideal per al desenvolupament d'algoritmes, modelatge i simulació en el processament d'imatges. La seua capacitat per a treballar amb matrius i programació orientada a matrius el fa perfecte per al TDI. Per aquestes raons, MATLAB serà utilitzat en aquest TFG, concretament, amb la versió 2023b.

3. TÈCNIQUES DE TDI

Comprendre els aspectes teòrics dels diversos mètodes de processament d'imatges no només és essencial per a una correcta aplicació d'aquestes tècniques, sinó també per a la innovació i l'optimització de solucions en projectes específics.

S'han seleccionat els procediments de TDI que es consideren fonamentals per a facilitar l'enteniment d'aquest projecte, aquest conjunt inclou la segmentació per a

[1] En el context de MATLAB, es defineix una *Toolbox* com una col·lecció de funcions i ferramentes addicionals com a extensió del programari, la qual cosa permet als usuaris dur a terme tasques especialitzades de forma més eficient i efectiva (Gonzalez, Woods i Eddins, 2004).

l'extracció de característiques rellevants, els descriptors per a la classificació d'aquestes característiques, les tècniques d'aprenentatge profund o *deep learning* per a tasques complexes com el reconeixement d'objectes, i la correlació, que s'usa per a la comparació i identificació de patrons.

Aleshores, aquest coneixement teòric és indispensable per comprendre les limitacions i els avantatges de cada mètode, facilitant així una elecció informada i una implementació més efectiva de cara al plantejament de l'aplicació; amb aquestes bases sòlides, serà senzill relacionar les funcions del generador de *moodboards* que es vol crear amb les diferents tècniques de TDI.

El present apartat proporcionarà una visió general dels principals mètodes de processament digital d'imatges, on cada secció estarà dedicada a un conjunt específic de tècniques, explicant no només els fonaments teòrics sinó també algunes de les aplicacions pràctiques i escenaris en què es poden utilitzar amb èxit.

3.1. Segmentació

La següent part es destina als mètodes de segmentació, que es corresponen amb la tercera etapa del procediment que s'il·lustra a la Figura 2 vista anteriorment, que també es pot considerar com a la primera fase d'anàlisi d'una imatge. Aquest grup de tècniques se situa en el nivell mitjà en l'escala de dificultat de processament.

Segmentar una figura significa subdividir-la en les distintes parts que la conformen. L'objectiu és identificar els grups de píxels que formen subconjunts o agrupacions dintre de la imatge. Hi ha mètodes més simples i d'altres més complexos per a realitzar aquesta separació d'objectes; a continuació, s'expliquen només aquells vists a les classes impartides durant el curs, el mecanisme dels quals es considera més bé senzill.

3.1.1. Mètodes de *thresholding*

La llindarització, més coneguda com a *thresholding* al camp del tractament digital de la imatge, permet segmentar una imatge fixant-se en els límits d'un llindar que s'estableix a l'histograma d'una imatge.

El mètode més bàsic és el de llindarització simple, que es realitza amb operacions píxel a píxel. Aquest, s'empra en els histogrames simples d'imatges binàries, o d'una banda (en escala de grisos), per a diferenciar un objecte del seu fons.

El procés és el següent: mitjançant un algoritme automàtic[2] o el criteri humà, se selecciona un llindar que divideix el fons de l'objecte a l'histograma, el qual conté un total de 256 valors, on el número 0 representa el color blanc i el 255 el color negre (també és normal emprar els valors 0 i 1 en lloc dels anteriors). Quan una imatge té menys nivells de gris, és més fàcil trobar el llindar correcte. Per això, és important reduir el soroll abans de segmentar la imatge. A la **Figura 3** es veu com el soroll fa difícil trobar el llindar adequat.

[2] Un dels algoritmes automàtics de selecció de llindar global més utilitzats és el d'Otsu, que s'implementa a MATLAB mitjançant el comandament *graythresh* (per a aprofundir, vegeu Magro, 2013).

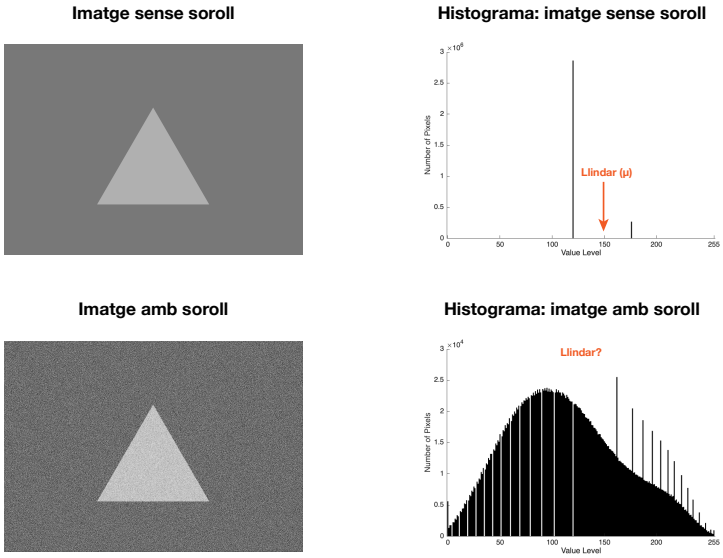


***Figura 3**. Mètode de segmentació per llindarització simple sense i amb soroll.*

La il·luminació també dificulta la llindarització. Si una imatge no està ben il·luminada, els nivells del seu histograma es superposen i és difícil trobar el llindar. La següent figura compara l'histograma d'una imatge sense soroll amb el d'una imatge mal il·luminada.

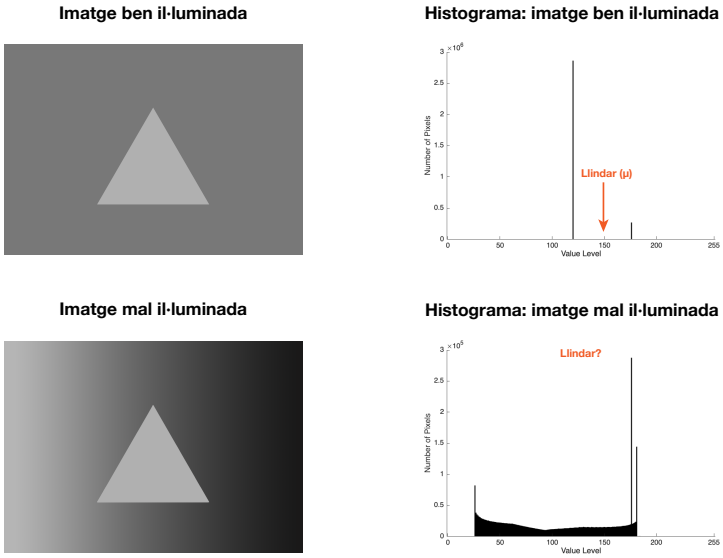


***Figura 4**. Mètode de segmentació per llindarització simple amb bona i mala il·luminació.*

En aquests casos, la solució és, o bé corregir la il·luminació estimant el seu fons; o bé establir un llindar adaptatiu que es moga depenent de les distintes zones amb diferent il·luminació; o bé fer ús de detectors de vores per a una ràpida segmentació (vegeu Sappa i Devy, 2001). Aquests, tan sols són els mètodes suggerits per a la correcció d'il·luminació, però existeixen molts processos recomanats per al tractament de cada imatge segons el seu punt de partida (Magro, 2013).

No obstant això, també es poden segmentar imatges amb múltiples components, com el color roig i blau, o la saturació i brillantor. En aquests casos, s'utilitzen histogrames 2D, que permeten una doble llindarització amb un llindar per cada component.

La llindarització 3D, per a imatges amb més de dos plans de color (com RGB), és més complexa. L'histograma és difícil de representar i sovint cal més d'un llindar per component, a més de tècniques de *clustering*.

Aquests mètodes s'usen per eliminar fons no desitjats, detectar moviment en càmeres de seguretat i identificar defectes en productes industrials.

3.1.2. Mètodes de *clustering*

Una altra forma de discernir les parts que conformen una imatge és mitjançant la clusterització, on existeix un problema d'agrupació i, per tant, s'ha de decidir a quin grup pertany cada píxel escollint una propietat similar comuna, com el color representatiu de cada conjunt.

Hi ha una gran multitud d'algoritmes automàtics d'agrupació de dades que s'encarreguen de fer-ho, però, en aquest context, només n'interessen dos, que són els estudiats a l'assignatura de Tractament digital d'imatge i vídeo: K-Means i Mean Shift. Es creu adequat analitzar aquests dos algoritmes en concret, a causa de la seua popularitat i efectivitat en el processament d'imatges i visió per computadora, junt amb la seua capacitat d'adaptació i millora d'altres algoritmes (Alorf, 2012).

Per un costat, K-Means tracta de dividir un conjunt de dades en K grups (clústers), on cada punt o píxel pertany al grup el centre del qual és més proper a ell.

L'algoritme comença seleccionant quants clústers hi ha (K), inicialitza el centre de cadascun de forma aleatòria i assigna a cada punt al centre del clúster més proper. Després, es fa una mena d'actualització, on es recalculen els centres com a la mitja dels punts, i s'iteren de nou tots els passos (menys el primer) fins a assolir la convergència (que no hi haja canvis significatius en els centres dels clústers). Aquesta explicació s'acompanya d'un esquema visual que recull tot el procés pas per pas per a una millor comprensió (**Figura 5**).

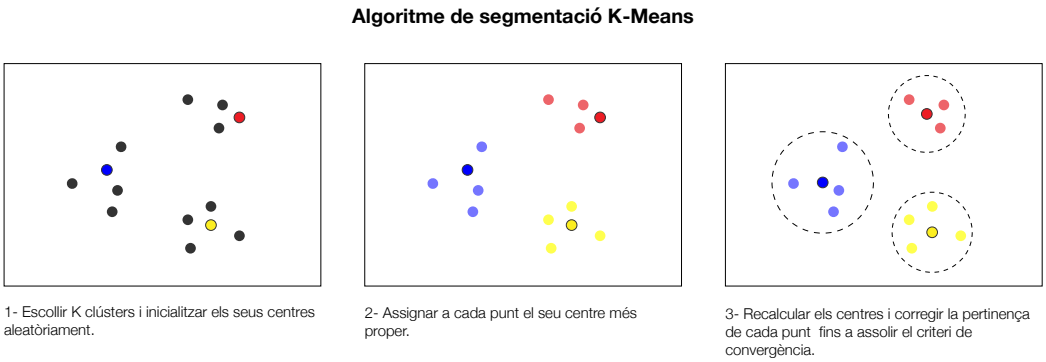


***Figura 5**. Procés de segmentació de l'algoritme K-Means pas a pas.*

D'altra banda, s'hi troba Mean-Shift, un algoritme que busca clústers molt densos en l'espai de color fins a trobar-ne el màxim local.

En lloc d'assignar punts a clústers com en K-Means, Mean Shift es basa en la idea de desplaçar una finestra de cerca (la mida de la qual s'ha d'establir prèviament) que calcule la mitja (com indica la paraula anglesa *mean*) dels punts que conté i oriente dita finestra cap al punt central calculat fins que aquest arribe a convergir en la regió de major densitat. En acabar el procés de desplaçament, es fusionen tots els punts d'un clúster que siguen pròxims al seu centre, de manera que cada punt formarà part del clúster al qual haja convergit després d'escombrar tot l'espai. Tanmateix, s'adjunta a la **Figura 6** una explicació gràfica del procés.

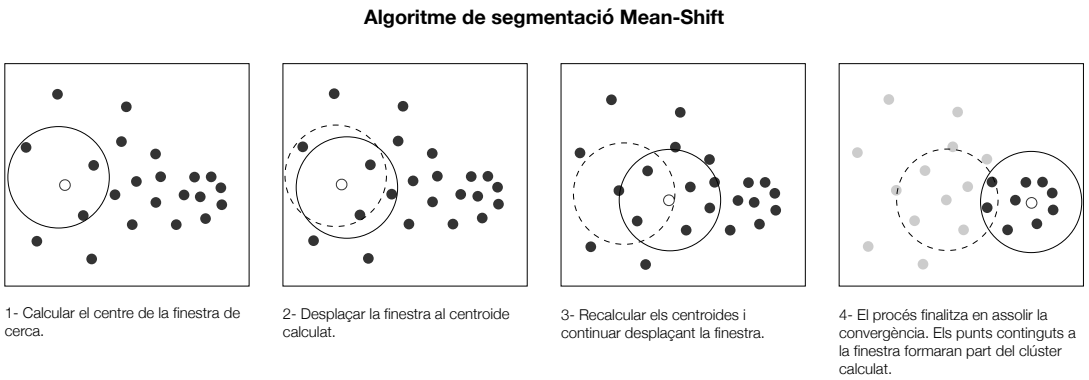


***Figura 6**. Procés de segmentació de l'algoritme Mean-Shift pas a pas.*

K-Means abasta una ampla varietat d'aplicacions; per exemple, la detecció de tumors en la medicina, o la identificació de defectes i classificació de materials per al control de qualitat de productes en la indústria. A banda d'aquests usos, Mean-Shift també és útil per a sistemes de seguiment, com el *tracking* d'objectes en vídeos, la detecció de persones en moviment en sistemes de vigilància, i més.

Les aplicacions anteriors denoten que un algoritme està a un nivell superior que l'altre. No obstant això, ambdós tenen avantatges i inconvenients. K-Means és senzill i computacionalment ràpid, però necessita que el valor de K siga prefixat i, a més a més, els resultats obtinguts són sensibles als *outliers*, que són aquells valors que disten bastant respecte la majoria dels valors del conjunt de dades. En canvi, amb Mean-Shift no cal prefixar K, tan sols és necessari determinar la dimensió de la finestra de cerca, l'algoritme no és sensible als *outliers* i és adaptable a qualsevol forma; els inconvenients són l'alta despesa computacional, la major complexitat i la dependència entre el resultat obtingut i la grandària de finestra establerta.

3.1.3. Segmentació orientada a regions

Un altre mètode de segmentació que implica operacions complexes és el de la segmentació orientada a regions. L'estratègia a emprar ací, es basa en els criteris de similitud i continuïtat dels píxels que formen una regió. Segons aquesta perspectiva, es considera que la imatge està formada per n regions separades en funció d'alguna propietat característica de cada zona (Platero, 2007).

Aquestes tècniques són iteratives, parteixen d'una situació inicial que van polint en cada pas. A diferència dels casos anteriors, la segmentació orientada a regions buscarà directament les diferents parts d'una imatge. A continuació, es mostren dos figures il·lustratives dels processos de *Region Growing* i *Split & Merge*.

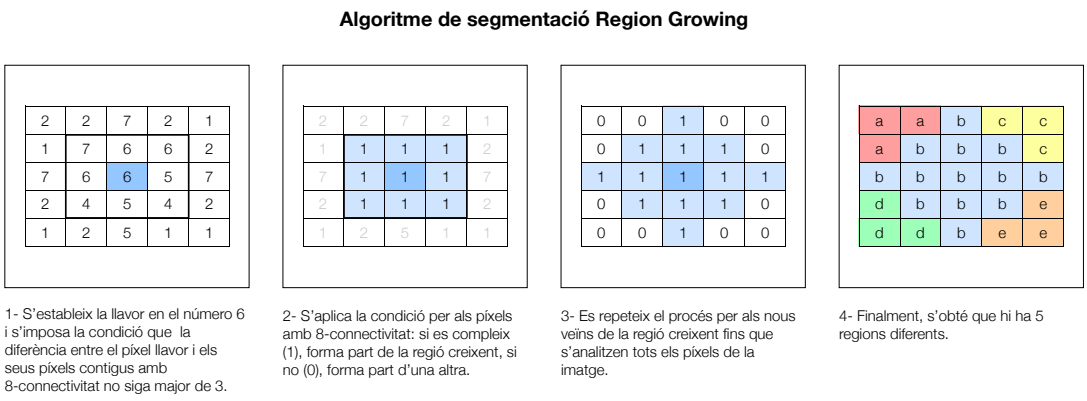


***Figura 7**. Procés de segmentació de l'algoritme Region Growing pas a pas.*

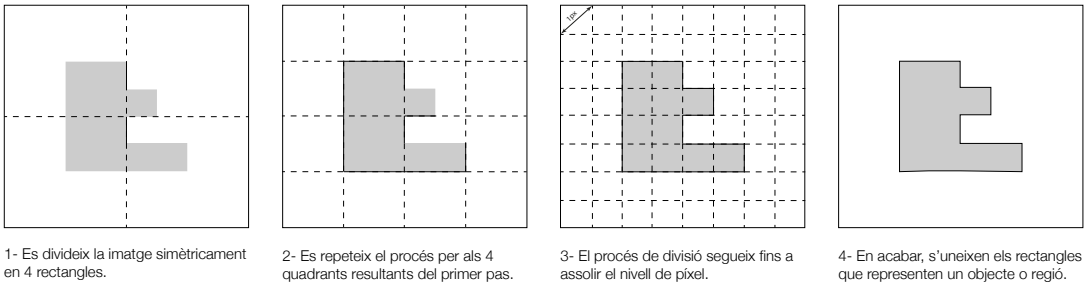


***Figura 8**. Procés de segmentació de l'algoritme Split & Merge pas a pas.*

3.2. Descriptors

Després del procés de segmentació d'imatges, ve el de descripció (**Figura 2**). Bàsicament, aquest procediment serveix per a poder distingir entre les diverses figures en què s'acaba de segmentar una imatge.

Els passos generals són l'extracció d'una sèrie de paràmetres característics dels objectes, denominats descriptors, que s'utilitzaran en la següent etapa del procés de visió artificial per a reconéixer i classificar objectes (com pot ser el cas del control de qualitat i la classificació de productes que van sobre una cinta mecànica en una fàbrica). Dits paràmetres s'emmagatzemen en forma de valors numèrics amb significat conegut.

Seguidament, d'entre la gran multitud de descriptors existents, es mostren per damunt aquells que es consideren més importants.

3.2.1. Descriptors de mida

Aquest tipus de descriptors és senzill, però només és útil si es coneix la distància als objectes per fixar-la prèviament. A més, depén de molts factors com la posició i orientació dels elements.

Els descriptors de mida mesuren característiques dels objectes per classificar-los. Per exemple, es poden extraure el perímetre, el diàmetre o l'àrea d'una figura (**Figura 9**).

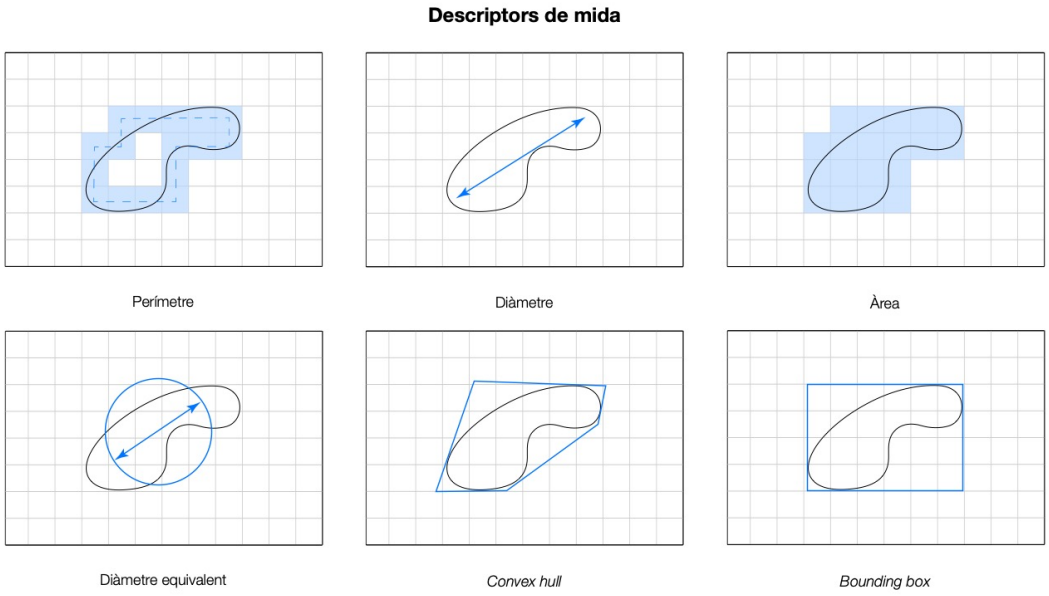


Figura 9. *Exemplificació dels paràmetres mesurats pels descriptors de mida.*

3.2.2. Descriptors de regió

Aquests són els més importants, ja que són invariants a la posició de l'objecte i descriuen la forma de l'àrea que ocupa l'objecte en qüestió.

A l'assignatura s'han vist els de compacitat, relació d'aspecte, moments centrals, moments de Hu i els perfils (**Figura 10**).

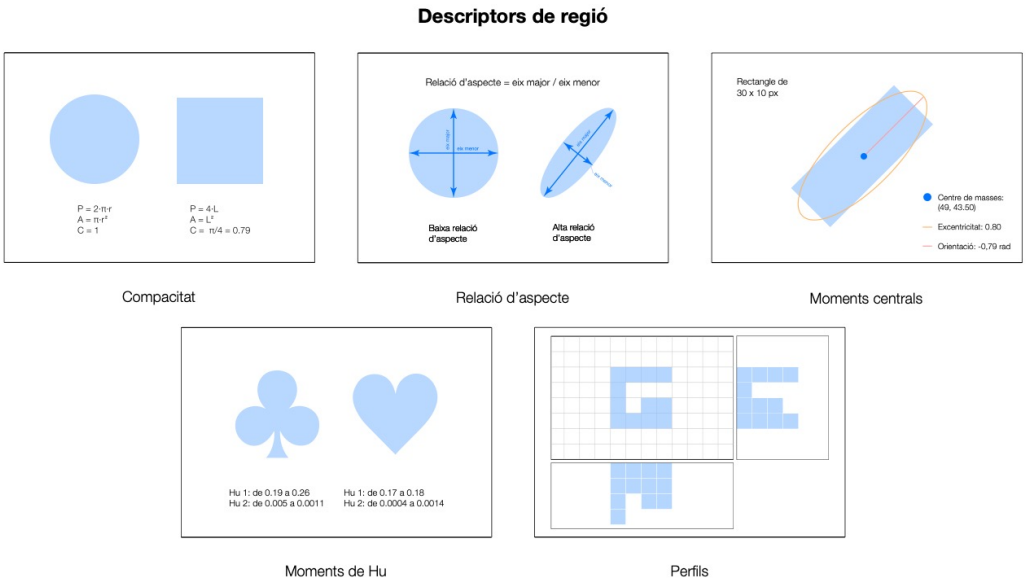


Figura 10. *Exemplificació de descriptors de regió.*

3.2.3. Descriptors avançats

Ara bé, és moment d'endinsar-se en els descriptors avançats, unes tècniques més complexes que serveixen per a detectar formes complicades que els anteriors descriptors no són capaços d'identificar.

Es tracta del reconeixement de formes orgàniques, com venen a ser les siluetes, parts o gestos del cos humà. A diferència dels anteriors mètodes, que solen identificar figures digitals o objectes simples, aquestes tècniques s'apliquen a escenes reals, per exemple, amb la detecció de vianants mitjançant càmeres de seguretat o el reconeixement de l'acte de somriure a través de les càmeres dels *smartphones*.

És cert que hi ha diversos mètodes dintre d'aquests descriptors, però sols es veuran els HoG i els HAAR per considerar-se més útils de cara a elaborar aquest TFG.

HoG significa *Histogram of Oriented Gradients*, i com el mateix nom assenyala, la seua funció és representar les orientacions dominants de les vores dels objectes per a identificar patrons. Dits patrons solen ser de persones, encara que també poden ser d'altres elements com caràcters, vehicles, i més.

El procés és clar, inicialment la imatge es divideix en NxN regions o blocs. Després, es calcula el gradient de cada píxel de cadascuna de les regions (per a més informació, vegeu Ramos, 2017). A continuació, es genera un histograma de les M orientacions resultants per bloc per a visualitzar aquelles predominants (solen haver-hi 8 orientacions per cadascun). Si la magnitud del gradient d'una regió és molt elevada, significa que la seua orientació està delimitant el contorn d'una silueta. En acabar, és necessari normalitzar les magnituds dels gradients per a reduir la sensibilitat a canvis d'il·luminació i contrast i per a millorar la robustesa dels descriptors. Per a la classificació de persones, s'empra l'algoritme d'aprenentatge SVM (*Support Vector Machine*). El procés del descriptor HoG es detalla pas a pas en la següent figura.

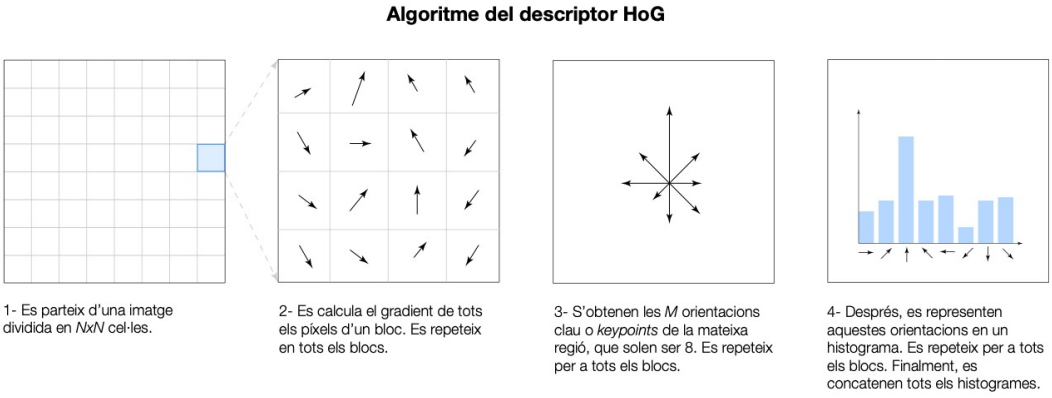


Figura 11. *Procés de l'algoritme del descriptor HoG.*

L'explicació en aquest apartat es queda curta per a com és en realitat el funcionament d'aquest algoritme. HoG és molt sensible als graus de suavitzat, per això, és recomanable reduir la imatge tot el possible abans d'emprar els descriptors, així, es podran detectar persones o elements tant propers com llunyans. Per a saber-ne més, convé consultar l'estudi titulat *Histograms of oriented gradients for human detection*, de Navneet Dalal i Bill Triggs (2005).

L'altre descriptor avançant del que es vol parlar és el de Haar-like features, també conegut per les sigles HAAR. Aquest, serveix per a detectar rostres calculant la

diferència mitjana de lluminositat entre regions rectangulars adjacents mitjançant la suma de píxels. A través dels canvis de lluminositat es poden localitzar patrons que representen faccions humanes. El seu gran avantatge és que l'algoritme és molt ràpid a l'hora de fer els càlculs malgrat tenir milers i milers de patrons de referència emmagatzemats.

El procediment seguit per a buscar rostres comença seleccionant un conjunt de característiques Haar distintives que capturen diferències en intensitats mitjanes entre regions d'una imatge. En acabant, s'empra l'algoritme AdaBoost per a aprendre un classificador fort combinant múltiples classificadors no tan potents basats en les característiques Haar seleccionades. Després, s'avalua la regió que s'està analitzant per a determinar si conté un rostre, o no, utilitzant el classificador fort aprés. Dita subfinestra es desplaça per tota la imatge per a buscar possibles rostres, i el procés d'avaluació es repeteix en diferents ubicacions i escales per detectar cares de diferents mides en la imatge.

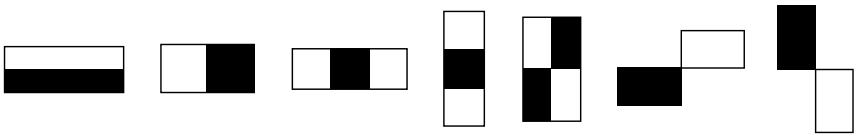


Figura 12. *Exemples d'atributs HAAR.*

Aquest algoritme també és més complex del que sembla. Podeu consultar més sobre el seu funcionament en l'acta de la conferència *Joint Haar-like features for face detection*, de Takeshi Mita, Toshimitsu Kaneko i Osamu Hori (2005).

3.3. *Deep learning*

L'última fase d'un sistema de visió artificial és la de reconeixement i interpretació (**Figura 2**). Ací, l'objectiu és delegar a una màquina el sentit humà de la visió; en alguns casos, aquests sistemes no són tan eficients com les persones, però en altres, superen les seues capacitats. Per exemple, actualment els sistemes de *deep learning* es poden emprar en l'automoció per a reconéixer senyals de trànsit i interpretar el seu significat a llarga distància, o en altres àmbits per al reconeixement de la identitat de les persones a través del seu rostre.

Pel que en aquest TFG concerneix, de l'última fase esmentada només es veuran les ferramentes d'alt nivell basades en *deep learning* (aprenentatge profund) i la correlació, a la qual es dedicarà el següent i últim apartat del marc teòric. El motiu és que, en haver-hi tantes tasques de reconeixement i interpretació, els sistemes es dissenyen enfocats a feines molt concretes perquè siguen efectius, per tant, cal avançar-se i centrar-se en aquells que es consideren més interessants i viables a l'hora d'implementar.

L'aprenentatge profund pretén proporcionar la categoria d'un objecte a partir dels descriptors obtinguts en la fase anterior. Per a comprendre el seu complex funcionament, aquest punt es divideix en diversos apartats, on cadascun presenta un concepte important en el *deep learning*.

3.3.1. Formes i classes

L'agrupació de múltiples descriptors en un mateix vector (*feature vector*) es coneix com a forma. Els trets d'objectes que guarda el vector poden ser, per exemple, el diàmetre, el perímetre, la relació d'aspecte, la compacitat...

Els objectes es classifiquen en funció de la forma de cadascun. Per a agrupar en un família diversos elements que comparteixen certes característiques, es fan servir les classes.

3.3.2. Procés de classificació

Prendre la decisió d'assignar una forma a una classe no és senzill. És per això que s'ha de recórrer a un conjunt d'entrenament.

Els conjunts d'entrenament són amplis grups de dades prèviament classificades mitjançant visió humana (per exemple, la classificació d'imatges d'animals). Aquests conjunts serveixen per a entrenar models o algoritmes d'aprenentatge perquè puguen operar de forma independent. Es coneix aquest procés d'entrenament com a *Machine Learning*.

Hi ha dues fases que componen el *Machine Learning*: l'entrenament i el testatge. Com que la fase d'entrenament estima la funció de predicció f que minimitza l'error del conjunt d'entrenament, interessarà donar-li les màximes dades possibles. D'altra banda, la fase de test aplica la funció f apresa per a classificar un conjunt de test i, així, avaluar si el model entrenat funciona correctament. S'ha de tenir clar que un conjunt de test no és el mateix que un conjunt d'entrenament; el conjunt de test és molt més reduït que el d'entrenament i les dades que se li donen també són diferents amb tal d'avaluar bé l'algoritme i corregir els errors necessaris.

S'ha mencionat la importància de tenir un gran *dataset,* però també és molt important seleccionar descriptors que siguen: discriminatius, senzills de calcular, invariants, robustos a les distorsions i el soroll (entre altres) i que els conjunts siguen tan incorrelats com siga possible.

Dit açò, el primer pas del procés de classificació és la selecció del model d'entrenament: paramètric o no paramètric (vegeu Montoro, 2015). Després, s'escull si l'entrenament és supervisat o no supervisat; en el cas del *deep learning*, es parla d'un entrenament no supervisat, que significa que no se li subministra inicialment cap categoria a la màquina. Finalment, amb la fase d'avaluació s'obtenen els resultats del conjunt de test.

3.3.3. Models de decisió: xarxes neuronals

Existeixen molts tipus de classificadors que prenen les decisions en el procés de classificació. No n'hi ha cap que siga millor que altre. Entre ells, es troben els models SVM i AdaBoost, mencionats en el punt 3.2.4. No obstant això, es considera irrellevant entrar en detall en els models de decisió, n'hi ha prou amb saber que el principal mètode emprat en el *deep learning* és el de xarxes neuronals (*Neural Networks*).

Aquestes xarxes pretenen imitar el comportament del cervell humà. La seua idea és definir una xarxa de neurones interconnectades, on cadascuna prendrà una decisió entre dues classes (o decidirà si una classe és correcta o no), a partir d'estímuls d'entrada, que són els descriptors dels objectes. El resultat de la decisió es traurà per les eixides de la xarxa neuronal. L'arquitectura d'una xarxa de neurones artificial es veu com la de la següent figura, que consta de diverses capes: capes d'entrada, capes ocultes (d'ací el concepte *deep*) i capes d'eixida.

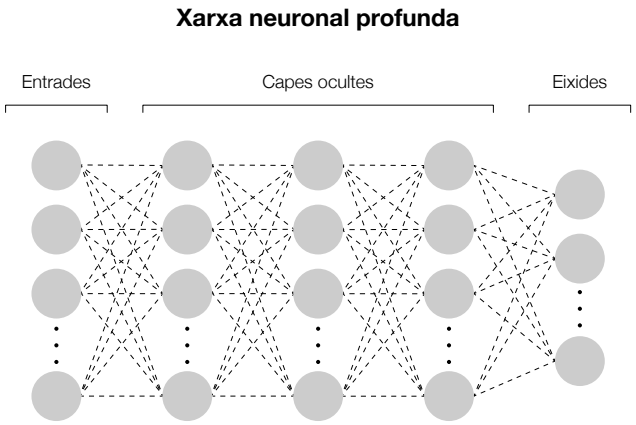


Figura 13. *Xarxa neuronal profunda típica.*

Hi ha tantes entrades com dimensions tinga el vector que representa el cas a classificar, i hi pot haver fins a milers de capes ocultes. Totes les unions entre neurones es multipliquen amb el que s'anomenen pesos; el valor d'aquests pesos determina la importància que té cada connexió, on 0 és el mínim valor (desactivat) i 1 és el màxim. El tipus de neurona artificial més utilitzat és el perceptró, que funciona com un discriminador lineal binari, és a dir, només classifica elements entre dos grups i tan sols ho pot fer dibuixant una línia recta entre ambdós grups, com es veu a la **Figura 14**. En cas que no es puga dibuixar una línia recta entre classes, es fan servir xarxes neuronals multicapa amb funcions d'activació no lineals.

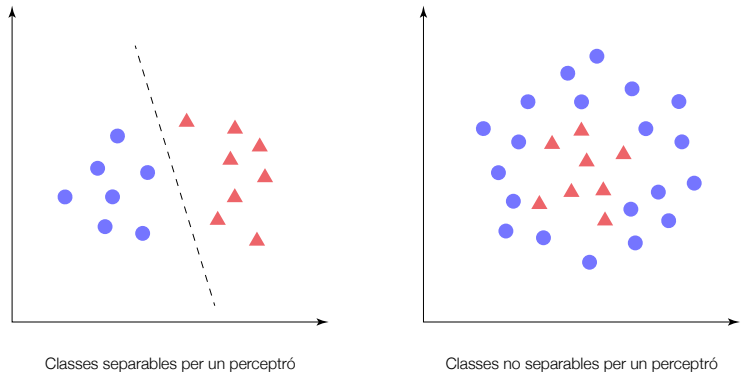


Figura 14. *Comportament del perceptró com a discriminador lineal binari.*

El truc del *deep learning* resideix en l'entrenament, que consisteix a calcular els pesos de cadascuna de les neurones a partir d'un conjunt d'entrenament existent. El

procediment s'anomena *backpropagation*, és un procés iteratiu de prova i error que ajusta els pesos d'una xarxa neuronal per a minimitzar l'error de classificació.

En aquest context, les xarxes neuronals convolucionals (CNN o R-CNN) juguen un paper fonamental. Les CNN convolucionen característiques apreses amb dades d'entrada i utilitzen capes convolucionals en 2D per a processar dades en dues dimensions, com ara imatges, ja que les CNN extrauen característiques directament de les imatges. Les característiques rellevants es van aprenent mentre la xarxa s'entrena amb un conjunt d'imatges. Aquest procés automatitzat d'extracció de característiques contribueix significativament a la precisió dels models de *deep learning* en tasques de classificació d'imatges. A més a més, les CNN es poden utilitzar per a classificar altres tipus de dades, com ara text escrit a mà.

Per a obtenir més informació sobre aquestes tècniques, consulteu l'apartat de *deep learning* a la pàgina oficial de MATLAB[3].

3.4. Correlació

Per a tancar el marc teòric, s'explica què és el concepte de la correlació en el camp del tractament digital de la imatge.

Segons la RAE (2014), el terme correlació significa correspondència o relació recíproca entre dues o més coses o sèries de coses. En aquest cas, el concepte fa referència a un operador que s'empra en les tècniques de Visió Artificial com a instrument per a cercar patrons continguts en una imatge –un procés també conegut com a *Pattern Matching*– (Platero, 2007). Aquesta tècnica té aplicacions en camps com el processament d'imatges, el reconeixement de veu i el reconeixement de patrons.

El *Pattern Matching* és una eina poderosa en l'anàlisi de dades que mesura la similitud entre dos senyals. Aquesta tècnica estadística determina la relació estreta entre ambdós mitjançant els coeficients de correlació creuada.

Per exemple, si una imatge es passa a escala de grisos, cada píxel ve representat per un valor, de manera que una imatge es pot emmagatzemar en forma de matriu. La correlació es calcula comparant els valors de dues matrius A i B. Es posa el supòsit què B és una imatge base i A és un retall de la imatge original B (**Figura 15**).

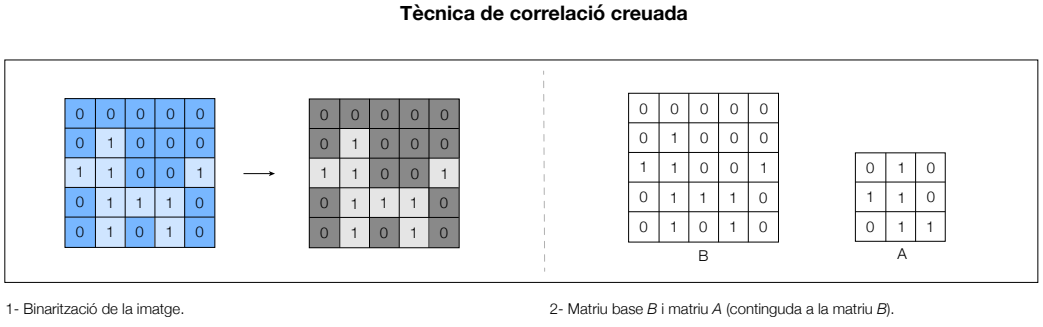


Figura 15. *Binarització i obtenció de dues matrius de píxels per a la correlació creuada.*

[3] The MathWorks Inc. (2024): *Introducción a Deep Learning*, Natick, Massachusetts: The MathWorks Inc., Disponible en <https://es.mathworks.com/discovery/deep-learning.html> [Consulta: 22-05-2024]

Per a comparar-les, se superposa la matriu A damunt de la matriu B i, seguint un ordre de coordenades que ve determinat per la cel·la del mig de la matriu A, els valors superposats es multipliquen i se sumen. En cada moviment d'A es fa el càlcul de les cel·les superposades, el resultat del qual es guarda en una nova matriu C de coeficients de correlació, i no en qualsevol casella, sinó en la corresponent a la posició central de la matriu A en cada desplaçament.

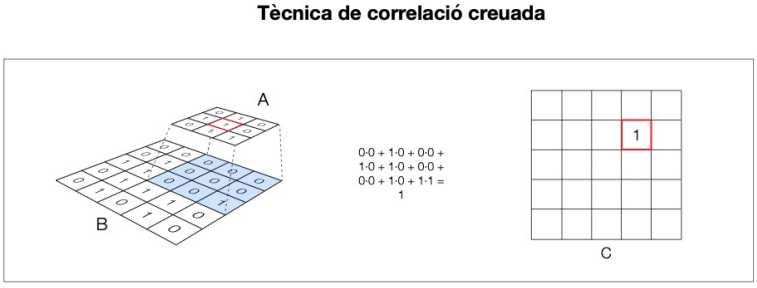


Figura 16. *Càlculs de la correlació creuada entre les matrius A i B.*

De la matriu C resultant, s'obté un valor màxim de correlació. D'aquesta manera, es pot veure quin grau de coincidència tenen les matrius A i B i en quina zona es dona aquesta similitud, que és el mateix que dir que dues imatges tenen els píxels molt semblants o iguals en una àrea determinada.

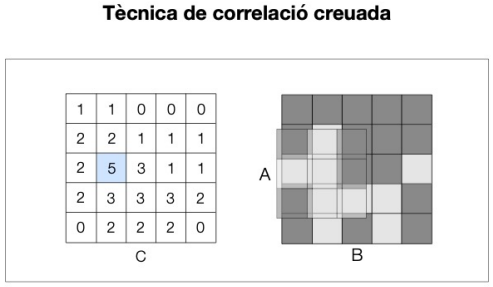


Figura 17. *Resultat de l'exemple de correlació creuada.*

La correlació ja explicada es coneix com a correlació creuada. El seu nom es deu a la comparació vista, on dos senyals interactuen de forma creuada mitjançant els desfasaments temporals. És la forma més bàsica de correlar i, com també s'ha vist, és útil per a localitzar una plantilla donada dins d'una imatge. Aquest tipus d'anàlisi es pot realitzar de forma senzilla al programa MATLAB, tenint en compte que totes les imatges emprades han de ser binàries. El desavantatge de la correlació creuada és que no està escalada, el que significa que els seus valors poden variar prou segons les amplituds dels senyals comparades. Vegeu un exemple més complex que l'anterior.

Es comparen dues matrius de píxels que difereixen en la intensitat dels píxels. Si no s'ajusten les variacions d'amplitud, els valors de la matriu de coeficients de correlació estaran distorsionats, la qual cosa dificultarà la interpretació dels resultats.

Figura 18. *Correlació creuada no normalitzada.*

Per a resoldre aquest problema, es fa ús de la correlació creuada normalitzada, que és independent del nivell de gris, sols té present la forma i permet tenir les dues variables en una mateixa escala. El que fa aquest tipus de correlada és ajustar els valors de correlació per a garantir que estiguen continguts dintre d'un rang consistent que sol oscil·lar entre -1 i 1, on 1 indica una coincidència perfecta, -1 una anticoincidència perfecta, i 0 cap correlació. Per aquest motiu, amb tal d'evitar errors de màxims, es prefereix fer servir la correlació creuada normalitzada a la creuada. Aplicant la següent fórmula (Vinuesa, 2016), s'obtindria el resultat de la **Figura 19** en comptes del de la **Figura 18**, el qual és molt més fàcil d'interpretar una vegada es tinguen tots els coeficients de la matriu C.

$$Coef.de\ correlació\ de\ Pearson\ = \frac{\sum(x_i - \bar{x}) \cdot (y_i - \bar{y})}{(N-1) \cdot s_x \cdot s_y}$$

Figura 19. *Correlació creuada normalitzada.*

A diferència de l'altra correlada, la normalitzada té en compte també l'energia de dos senyals per a veure si, realment, són semblants o no; d'aquesta manera, sí o sí, s'obtindran uns màxims de correlació correctes, fins i tot amb dos senyals iguals o semblants que estiguen invertides.

Els valors resultants d'aquestes operacions s'emmagatzemen en escalars numèrics o en vectors de distints graus de complexitat; aquest és un factor a tenir en compte de cara a escollir uns mètodes o uns altres.

4. PLANTEJAMENT DE L'APLICACIÓ

4.1. Relacionar un *moodborad* amb tècniques de TDI

Ara que ja es coneixen els mètodes que es fan servir al tractament digital de la imatge i algunes de les seues aplicacions, serà molt fàcil plantejar quines tècniques tenen cabuda a l'hora de crear un *moodboard*.

Recapitulant, un *moodboard* inclou elements com imatges, textos, colors, i materials, que es disposen en un mateix espai de manera visual. Amb aquesta informació, es relaciona el tauler d'inspiració amb el TDI de la següent forma:

- Una de les característiques que no pot faltar en cap *moodboard* és la paleta de colors que representa les tonalitats generals del conjunt d'imatges. Per tant, es pot relacionar aquesta funció amb el *clustering*, que és molt efectiu per a detectar zones de píxels amb valors semblants i obtenir així els colors més representatius de cada àrea, que ve a ser la paleta cromàtica.

- D'altra banda, és comú inserir textos que representen emocions, sentiments o suggeriments. MATLAB també és capaç de treballar fàcilment amb textos, però per a afegir-li un component relacionat amb l'àrea del TDI, es planteja la possibilitat de detectar tipografies a partir d'imatges donades i que l'aplicació imprimisca els textos que l'usuari desitge amb la tipografia detectada. Aquest tipus de detecció es pot fer amb descriptors o amb processos de correlació.

- Com s'exemplifica a la **Figura 1**, alguns *moodboards* presenten retalls de persones perquè hi ha casos on no interessa mostrar el fons de les imatges. Per tant, es proposa detectar persones en imatges per a retallar la seua silueta i afegir-les al collage que es vol crear, aportant així formes més orgàniques i una composició més interessant. En l'àmbit del TDI, hi ha diverses possibilitats per a identificar contorns amb mètodes de segmentació, amb descriptors o amb *deep learning*, com s'ha vist als punts 3.1, 3.2 i 3.3.

4.2. Proves inicials amb MATLAB

Amb l'objectiu d'acotar el plantejament anterior, es fan unes proves inicials amb MATLAB per a fer-se una idea de quines funcions i quines tècniques en concret poden realitzar les accions del generador de *moodboards*.

4.2.1. Generació de paletes cromàtiques

Per a l'obtenció de la paleta de colors del tauler d'inspiració, es pretén aprofitar el generador de paletes creat com a projecte final de l'assignatura referent a aquest TFG, realitzant les modificacions pertinents al codi.

Una paleta de cromàtica representa aquells colors dominants en una imatge. La identificació d'aquests colors es pot comparar amb la cerca de les distintes regions de color dominants dintre de la imatge. Segons la informació vista al marc teòric, el més viable per a dividir una figura en distintes àrees que la conformen és emprar tècniques de segmentació, que, com s'ha explicat, troben els grups de píxels que formen subconjunts o agrupacions.

De primeres, es descarten els mètodes de *thresholding*, ja que treballen només amb imatges binàries i no amb colors RGB, a part què la segmentació que realitza sol localitzar tan sols dos subconjunts de píxels, i una paleta sol tenir entre 3 i 5 colors. Com a conseqüència, es podran emprar o bé mètodes de segmentació orientada a regions, o bé per *clustering*; ambdós proporcionen resultats similars.

Pel que fa a la segmentació orientada a regions, els seus mètodes poden oferir una segmentació ben precisa gràcies a la iteració dels seus processos, però aquest mateix aspecte fa que tinguen major complexitat de càlcul; al repetir moltes vegades el mateix procés fins a refinar els resultats finals, alenteixen el temps de les operacions i gasten més recursos.

Es decideix, per tant, que la millor opció per a segmentar regions de colors és el *clustering*, ja que els resultats són molt útils i les tècniques que empra no ocupen tanta despesa computacional, són ràpides i simples. Aquesta metodologia agrupa de manera efectiva els píxels amb colors similars i facilita el càlcul de la mitjana de color per a cada grup, que és just el que es necessita. A diferència de la segmentació orientada a regions, no s'ha de definir cap llavor, la qual cosa dota aquests processos d'una major adaptabilitat.

D'entre les tècniques de *clustering* K-Means i Mean Shift estudiades, es tria utilitzar K-Means per ser la més senzilla i ràpida de les dues en el procés de segmentació, ja que el primer objectiu en el desenvolupament del generador de paletes de color és simplement trobar les regions de diferents colors. Amb K-Means, s'aconseguirà fer la separació de les àrees esmentades per a posteriorment ordenar-les de major a menor nombre de píxels, és a dir, segons la seua predominança. Un punt a tenir en compte és la definició inicial del nombre de clústers; en aquest cas, no és un problema, ja que es vol definir el nombre de colors de la paleta abans de generar-la. A més a més, com que el resultat obtingut varia amb cada inicialització dels clústers, açò és interessant per a regenerar la paleta i disposar de diverses opcions. L'únic inconvenient és que no detecta formes irregulars: K-Means assumeix que els clústers són esfèrics, la qual cosa pot no ser òptima si els colors de la imatge tenen una distribució complexa, i açò amb Mean-Shift no suposa cap problema.

Per al projecte del generador de paletes, es va establir un rang d'entre 3 i 7 colors, d'entre els quals l'usuari ha d'escollir quants en vol. Amb els clústers inicialitzats i classificats per mida descendent, el següent pas és calcular la mitjana de cada regió; aquest valor representa el codi de color RGB de cada clúster, és a dir, cada color de la paleta cromàtica. Finalment, s'associa cadascun d'aquests valors a una gràfica o figura que mostra per pantalla el conjunt de colors.

Aquesta aplicació es va dissenyar i implementar exitosament amb l'entorn de desenvolupament per a aplicacions MATLAB App Designer. L'aspecte final de la interfície del generador de paletes de colors realitzat junt amb Álvaro Herrera es mostra a la **Figura 20**, on es pot observar mitjançant un exemple la funció que realitza.

Figura 20. *Interfície de l'aplicació generadora de paletes de color.*

En addició, s'adjunta una figura que il·lustra el funcionament de l'algoritme de l'aplicació a través d'un esquema explicatiu.

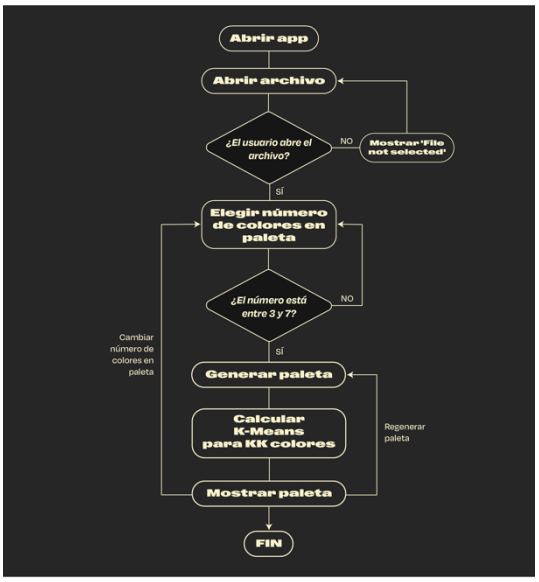


Figura 21. *Algoritme de funcionament del generador de paletes cromàtiques.*

4.2.2. Identificació de tipografies

Les proves per a la identificació de tipografies es fa amb el Live Script (o Live Editor) de MATLAB, que és un document on es poden implementar demostracions interactives.

Al punt 4.1 s'ha plantejat la possibilitat de reconéixer tipografies a través de dos mètodes del tractament digital de la imatge: descriptors o correlació. No s'ha plantejat l'opció de fer servir la segmentació, ja que aquesta sols separa uns objectes d'altres i no analitza les formes que els componen, però cal esmentar que pot ser útil en cas de tenir un fons poc uniforme on siga difícil discernir-ne el text. D'aquesta manera, es podria separar el fons del text i facilitar l'estudi posterior.

Dit açò, es planteja l'ús de descriptors per a identificar tipografies a partir d'imatges. Amb els descriptors de mida i de regió, es pot caracteritzar la forma i proporció de les lletres, però aquestes dades no sempre són fiables, ja que molts dels descriptors no són invariants a transformacions i la informació que proporcionen no és determinant a l'hora de comparar entre dos tipus de fonts. Potser els descriptors avançats sí que són capaços d'identificar els patrons específics de les tipografies, però es considera que hi ha un mètode més eficaç i precís per a fer aquesta tasca: la correlació.

Com bé s'ha assenyalat al punt 3.4, la correlació és útil per a localitzar una plantilla donada dintre d'una imatge, comparant directament una imatge d'un text determinat amb les imatges de les tipografies de referència, identificant així les coincidències entre elles. El gran benefici que diferencia aquest mètode dels descriptors és que no importa la mida del text de la imatge proporcionada, no han de ser iguals necessàriament; aquest aspecte fa que la correlació siga robusta. L'altre gran avantatge és que es tracta d'un procés simple d'implementar, com s'ha pogut apreciar a les **Figures 15, 16 i 17**, de manera que només amb el valor màxim dels coeficients de correlació ja es pot obtenir la tipografia més semblant.

Al capdavall, queda fer unes proves amb l'editor Live de MATLAB i concloure el codi aproximat per a la identificació de fonts. Inicialment, s'intenten correlar diverses imatges de text amb els comandos `xcorr2`, `normxcorr2` i `corr2`.

- Proves inicials amb `xcorr2`:

La funció xcorr2 només és eficaç a l'hora d'extraure els coeficients de correlació entre dos senyals, ja que calcula la correlació creuada per a diverses posicions relatives.

Cal esmentar que, a diferència de `xcorr`, el comando `xcorr2` serveix per a mirar la correlació creuada entre senyals bidimensionals i no unidimensionals (com és el cas de `xcorr`); com que es va a treballar amb imatges (que són senyals bidimensionals), cal emprar la correlació creuada amb `xcorr2`.

A continuació, s'adjunta el codi emprat per a provar un exemple amb la funció de correlació `xcorr2`, on es proporcionen dues imatges amb la paraula "Prueba" escrita amb la mateixa font i desplaçada en els eixos x i y. El resultat d'aquest test es pot apreciar a la **Figura 23**.

Exemple de xcorr2 de Matlab

```
% Importe imatges
SinSerifa_M1 = imread("/Users/macbookpro/Documents/MATLAB/Examples/R2023b/images/UseCrossCorrelationToFindTemplateInImageExample/SinSerifa_M1.png");
SinSerifa_M2 = imread("/Users/macbookpro/Documents/MATLAB/Examples/R2023b/images/UseCrossCorrelationToFindTemplateInImageExample/SinSerifa_M2.png");

% i les mostre
imshow(SinSerifa_M1);
title("Imagen M1");
imshow(SinSerifa_M2);
title("Imagen M2");

% Pasem a double (tindrem més capacitat)
M1double = double(SinSerifa_M1);
M2double = double(SinSerifa_M2);

% i normalitzem, així evitem perdre informació si processem
M1 = M1double./max(M1double(:));
M2 = M2double./max(M2double(:));

% Passem a escala de grisos per a poder utilitzar xcorr2 (2 dimensions)
M1g = rgb2gray(M1);
M2g = rgb2gray(M2);

% Correlem les matrius de les dues imatges
corr = xcorr2(M2g,M1g);

% Extrac el número màxim (y) i l'índex (x) de corr
[y,x] = max(corr(:));

% Dibuixe la correlada i localitze el seu máxim
figure
plot(corr(:))
title('Correlada creuada')
hold on
plot(x,y,'o')
hold off
text(x*1.05,y*1.05,'Maximo')
```

Figura 22. *Codi de prova amb la funció `xcorr2`.*

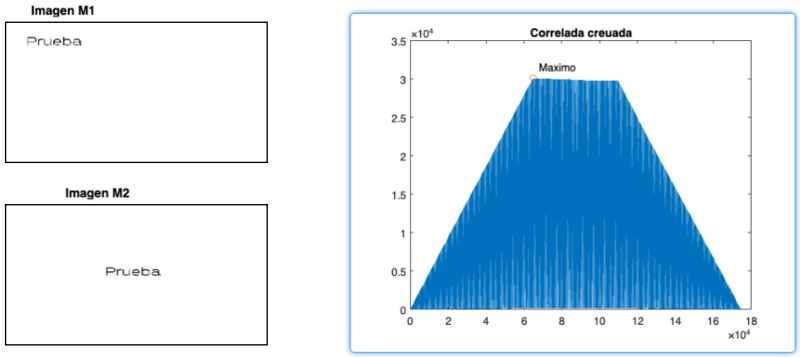


Figura 23. *Resultat de la prova amb la funció `xcorr2`.*

El problema amb `xcorr2`, a més de no proporcionar un valor global de correlació, és la seua despesa computacional. Generar una matriu de correlació creuada en dues dimensions és un procés que implica fer càlculs per a moltes posicions relatives

entre les imatges. En tornar un vector de dues variables que poden ser d'igual o distinta longitud (no com `corr2`, que torna un escalar numèric, com es veurà més endavant), el procés s'alenteix moltíssim en haver de generar vectors molt grans, ja que si ambdues imatges tenen la mateixa mida N, el mètode torna un vector de mida $2N - 1$.

- Proves inicials amb `normxcorr2`:

Després, s'intenten correlar diverses imatges de text amb el comando `normxcorr2`. En el seu cas, es calcula la correlada normalitzada per a corregir canvis d'intensitat lumínica, la qual cosa aporta major robustesa. No obstant això, es necessita que la imatge base siga més gran que la d'entrada perquè la correlació siga significativa, el qual és un gran inconvenient a l'hora d'introduir les imatges en MATLAB.

Si es comparen dues imatges amb la paraula "Prueba" amb la mateixa tipografia i angulació (però no la mateixa mida), els resultats són efectius. El patró del text s'identifica a la imatge base (com indica el rectangle blau de la **Figura 25**). Aquest és el codi de la correlació (**Figura 24**).

Prova correlació amb mateix text i posició ✓

```
Serifa = im2gray(imread('Serifa.png'));
Prueba2= im2gray(imread("Serifa_Recta.png"));
montage({Prueba2,Serifa})
c2 = normxcorr2(Serifa,Prueba2);
surf(c2)
shading flat
[ypeak2,xpeak2] = find(c2==max(c2(:)));
yoffSet2 = ypeak2-size(Serifa,1);
xoffSet2 = xpeak2-size(Serifa,2);
imshow(Prueba2)
drawrectangle(gca,'Position',[xoffSet2,yoffSet2,size(Serifa,2),size(Serifa,1)], ...
    'FaceAlpha',0);
```

Figura 24. *Codi de prova amb la funció* `normxcorr2`*.*

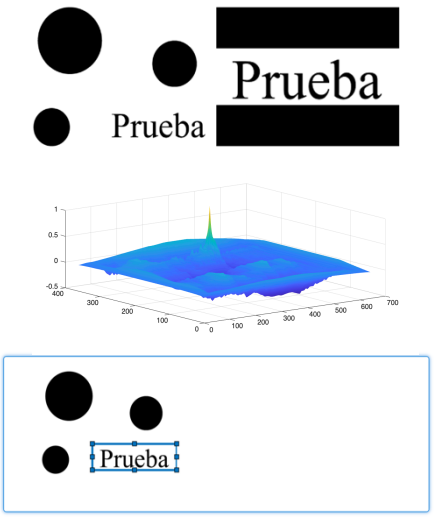


Figura 25. *Resultat de la prova amb la funció* `normxcorr2`*.*

Emprant el mateix codi que a la **Figura 24**, si l'angulació del text de la imatge més petita varia, el resultat ja no és eficient, com es veu a la **Figura 26**.

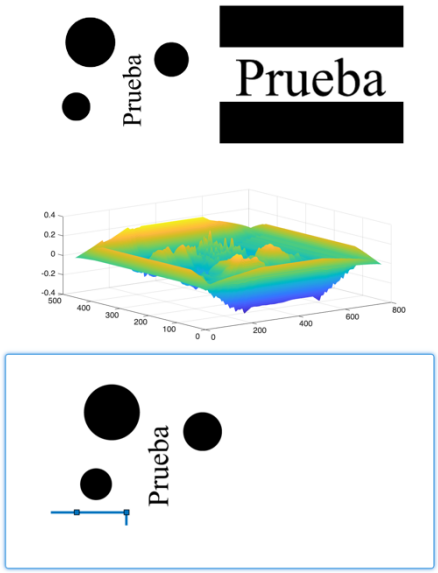


Figura 26. *Resultat de la prova amb la funció* `normxcorr2` *per a text rotat.*

Canviar el color del fons no influeix, com s'observa al següent exemple, ja que el comando es fixa en el fet que els valors dels píxels del text siguen semblants.

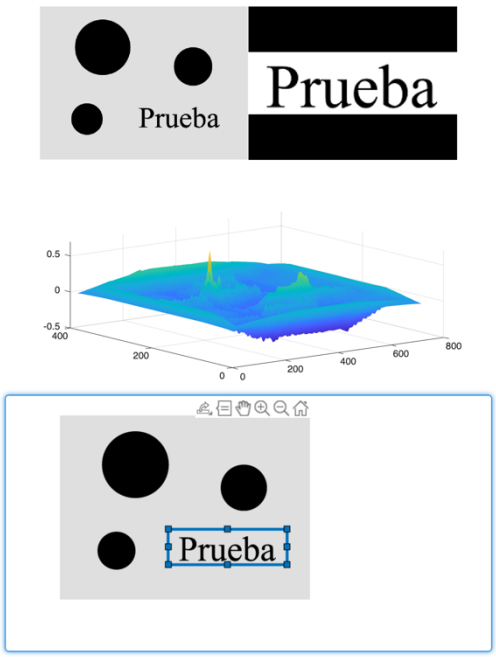


Figura 27. *Resultat de la prova amb la funció* `normxcorr2` *per a fons de distint color.*

Això sí, si es canvia també el color del text, la correlació no funciona per a `normxcorr2`. Com que no es tenen intensitats de píxel semblants, els dos textos no guarden correlació segons aquesta funció.

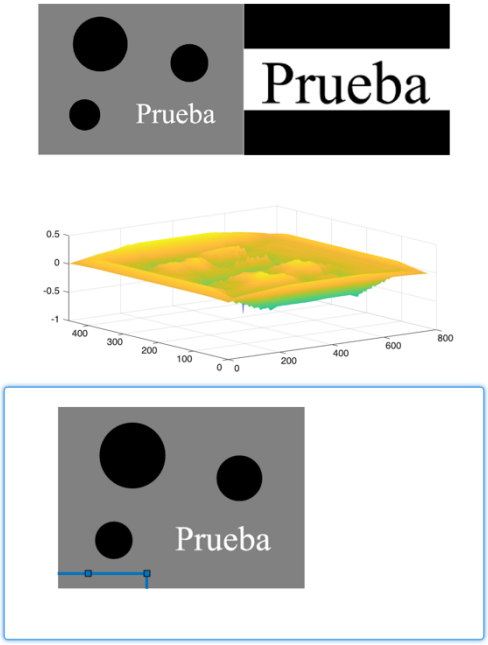


Figura 28. *Resultat de la prova amb la funció* `normxcorr2` *per a fons i text de distint color.*

També, es prova si la funció identifica el text donat amb una tipografia que no és la mateixa, però és molt semblant, i el resultat també és exitós.

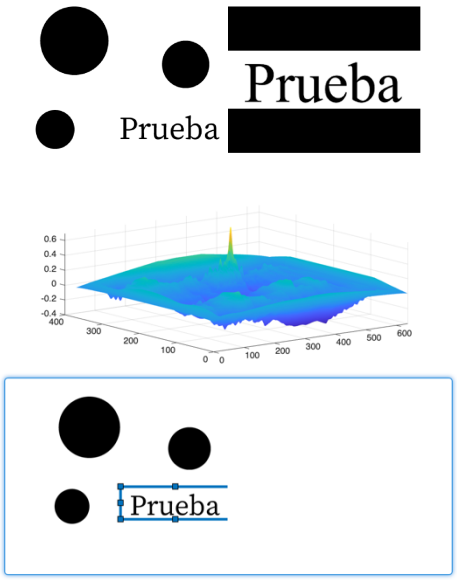


Figura 29. *Resultat de la prova amb la funció* `normxcorr2` *per a dues tipografies distintes.*

Finalment, es posa a prova `normxcorr2` col·locant 4 tipografies bastant distintes entre elles per a veure si identifica la de referència.

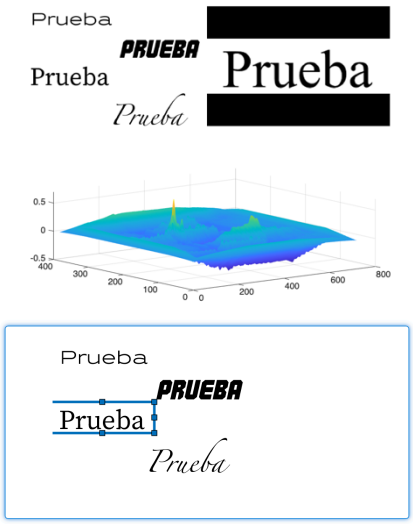


Figura 30. *Resultat de la prova amb la funció* `normxcorr2` *per a quatre tipografies distintes.*

Aquesta funció està dissenyada per a trobar la correlació normalitzada entre una plantilla bidimensional i una imatge, retornant una matriu que mostra com la plantilla es correlaciona amb diferents regions de la imatge. Com l'objectiu és determinar si existeix una correlació global entre dues imatges, el resultat que proporciona tampoc acaba de ser útil, perquè no es dona un valor general. Per tant, `normxcorr2` serveix per a identificar la posició òptima, però no per a obtenir una mesura única de similitud global.

- Proves inicials amb `corr2`:

En substitució de `normxcorr2`, es prova amb `corr2`, que funciona raonablement bé per al que es vol assolir.

Com s'ha dit, la intenció és obtenir una mesura única i directa de la similitud general. La funció `corr2` calcula el coeficient de correlació de Pearson, donant així un valor escalar entre `0` i `1` amb el qual es pot realitzar fàcilment la comparació. D'aquesta manera, es troba una eina més eficient i fàcil d'interpretar per avaluar la similitud global, ja que no s'afegeix la complexitat addicional de la normalització que té `normxcorr2`. El següent exemple ho il·lustra perfectament.

```
% Es carrega una imatge de la paraula "Prueba" amb dos tipus de fonts.

    Serifa = im2gray(imread('Serifa.png'));
    Decorativa = im2gray(imread('Decorativa.png'));

% S'agafa la mida del text de la imatge 'Serifa', amb el qual s'ajusta la
% segona imatge.

    tamtipo = size(Serifa);
    Decorativa = imresize(Decorativa,tamtipo);

% Es mostren ambdues tipografies comparades.

    montage({Decorativa,Serifa})

% Es mostra la tipografia 'Serifa' comparada amb ella mateixa.

    montage({Serifa,Serifa})

% Es calcula la correlació entre les dues fonts distintes.

    c1 = corr2(Serifa, Decorativa);

% Es calcula la correlació de font 'Serifa' amb ella mateixa.

    c2 = corr2(Serifa, Serifa);

% Si 'Serifa' se sembla més a ella mateixa abansa que a 'Decorativa',
% s'escriu per pantalla.

    if c2 > c1
        fprintf('La tipografia Serifa és més semblant a la Serifa, i el seu grau de correlació és de %.2f front a %.2f amb la tipografia Decorativa', c2, c1);
    else
        fprintf('La tipografia Serifa és més semblant a la Decorativa');
    end
```

Figura 31. *Codi de prova amb la funció* `corr2`.

Finalment, es conclou que aquesta és la millor opció per a identificar tipografies a partir d'una imatge donada amb un text. Com s'aprecia a la **Figura 32**, els valors de correlació són molt clars i fàcils d'interpretar. Si açò s'aplica a la llista de fonts d'un ordinador, es podria identificar la tipografia més semblant respecte a la que un usuari proporcione a través d'una imatge.

Figura 32. *Resultat de la prova amb la funció* `corr2`*.*

Amb la funció clara, es carrega al *script* una imatge amb un text amb una tipografia determinada. Per a detectar què posa en la imatge, s'empra l'Optical Character Recognition (OCR), que s'usa dins del camp de la morfologia, una ferramenta matemàtica utilitzada per a extraure components d'una imatge per a treballar amb la seua forma (o bé abans de la segmentació o bé després, en la fase de reconeixement i classificació, on és comú fer-ho servir per al reconeixement òptic de caràcters). Perquè la detecció siga més acurada, es busca la regió d'interés (ROI) i s'indica a una figura amb *bounding boxes*. En acabat, se li donen aquestes coordenades a l'OCR perquè identifique les paraules, de les quals n'extrau una, ajustant la imatge donada a la mida de la bounding box de la paraula seleccionada per l'OCR. Per a la detecció de text s'empra el comando `detectTextCRAFT`, una funció de MATLAB que detecta texts en imatges utilitzant el model de *deep learning* CRAFT (Character Region Awareness for Text detection).

A continuació, es passa la imatge retallada a escala de grisos (amb `im2gray`) i s'inverteix. Tot seguit, s'obté la llista de fonts de l'ordinador que s'estiga emprant (amb `listfonts`) i es trau la seua longitud; en aquest cas, la llista de tipografies té 394 fonts. La prova es basarà a anar comparant la imatge retallada amb totes les tipografies de la llista mitjançant un bucle, i buscar aquella que siga més pareguda fixant-se en el coeficient de correlació (amb `corr2`). El major coeficient es guardarà en una variable i el nom de la seua respectiva tipografia també. Finalment, es mostraran juntes la imatge del text seleccionat per l'OCR i la imatge de la tipografia que es considera més semblant, la qual s'obté del bucle. Tot aquest procés s'il·lustra a la **Figura 33**, on apareixen els comandos esmentats, entre altres.

```
% Es carrega la imatge donada

    textimg = imread('Serifa.png');

% Es detecta la posició del text en la imatge i s'indica amb un rectangle

    bboxes = detectTextCRAFT(textimg); % Es la 'Region Of Interest' (roi)
    Iout = insertShape(double(textimg),"rectangle",bboxes,LineWidth=3);
    figure
    imshow(Iout)

% Es crida a l'OCR i se li dona la ubiació de les paraules

    ocrResults = ocr(textimg,bboxes,'TextLayout', 'block');

% S'extrau el text trobat

    words = ocrResults(:).TextLines % Línies de text reconegudes per l'OCR
    newbbox = ocrResults(:).TextLineBoundingBoxes; % Coordenades de les bounding boxes

% S'ajusta la imatge donada a la mida de la bounding box

    textimgcrop=imcrop(textimg,newbbox);

% Es pasa la imatge retallada a escala de grisos i s'inverteix

    Tipografia = 255-im2gray(textimgcrop); % S'inverteix la imatge (fons negre i figura en blanc)
    tamtipo = size(Tipografia); % S'agafa la mida del text de la imatge retallada donada

% S'extrau la llista de fonts de l'ordinador de l'usuari

    l = listfonts;
    numfonts = length(l); % Número de tipografies

    maxcoef = 0; % Declaració variable per a emmagatzemar el coeficient de major valor
    figura = figure(1); % Nova figura

% Bucle per a comparar la imatge retallada donada amb totes les tipografies de la llista

    for ii = 1:numfonts
        clf(figura);  % Es borra la figura anterior
        text(0, 0, words,'FontSize', 36,'FontName',l{ii}) % Escriure el text detectat per la OCR amb la tipografia ii
        ax = gca;  % Eliminar els eixos
        ax.Visible = 'off';
        exportgraphics(figura,'Prueba.png') % Exportar la figura en una imatge .PNG
        Prueba = 255-im2gray(imread("Prueba.png")); % Importar la imatge en negatiu
        Prueba = imresize(Prueba,tamtipo); % Ajustar a la mida de la imatge donada inicialment per l'usuari
        coef = corr2(Prueba,Tipografia); % Càlcul del coeficient de correlació d'ambdues imatges
        if coef > maxcoef
           maxcoef = coef; % Actualització del coeficient màxim
           maxindex = ii; % Actualització de la posició de la llista del coeficient màxim
           imwrite(Prueba,'MillorTipo.png') % Es guarda la imatge del text més semblant amb el nom 'MillorTipo.png'
        end
    end

    close(figura)

% Es visualitzen ambdues fonts juntes

    Millortipo = imread("MillorTipo.png"); % S'obri la imatge de la tipografia més semblant
    figure, montage({Millortipo,Tipografia}) % Es mostren ambdues fonts juntes
    title(['Tipografia més semblant: ' l{maxindex}]) % S'escriu el nom de la tipografia a tall de títol
```

Figura 33. *Codi de prova d'identificació de tipografies.*

Aquest mateix *script* es prova substituint la funció `corr2` per les altres plantejades inicialment per a comprovar que, efectivament, la millor opció és l'esmentada.

Generalment, els resultats són molt semblants a la tipografia donada, com es veu a la **Figura 34**. Per tant, es té un mètode que funciona raonablement bé per al que es vol aconseguir, i aquest serà el que s'inclourà en l'aplicació.

Figura 34. *Resultat efectiu d'identificació de tipografies.*

4.2.3. Detecció de persones

Les imatges d'escenes reals són molt complexes a causa de la seua variant il·luminació, els diferents fons i la varietat de persones, per tant, no es pot fer ús del mètode de llindarització per a la segmentació de persones.

Dit açò, s'han de buscar altres mètodes que no depenguen dels anteriors paràmetres per a identificar persones a MATLAB, com per exemple, els descriptors avançats de HoG (punt 3.2.4). Aquests simplifiquen molt la representació d'una imatge, de manera que s'estalvia càlcul computacional al no fer ús d'informació innecessària. En comparar dues imatges per a identificar elements, és innecessari tenir en compte certs paràmetres que varien entres elles, com el fons, el color o les expressions facials, que només condueixen a tenir soroll. Per tant, HoG és perfecte per a guardar allò imprescindible i depreciar allò redundant.

Tot i això, cal destacar el gran avantatge del *deep learning* (punt 3.3), que demostren una excel·lent funcionalitat. Pel fet de ser més fàcils d'implementar i proporcionar resultats més precisos i eficients en comparació amb els descriptors HoG, es prefereix fer-ne ús d'elles.

En concret, s'empraran xarxes neuronals convolucionals basades en la regió (R-CNN). Per al cas, les R-CNN segmenten de forma individual cada individu que apareix en una imatge donada. Per a provar aquest tipus de model de *deep learning* s'ha obtingut un fitxer de MATLAB que conté la CNN per a la segmentació de persones i automòbils ("maskrcnn_object_person_car_v2.mat"). Gràcies a aquesta CNN, es té una base per a la detecció i extracció de persones en el *moodboard*.

D'altra banda, una vegada identificada una o diverses persones en una imatge, faltaria eliminar-ne el fons i exportar les imatges en PNG, de forma que el fons es quede en píxels transparents. Per a fer-ho, s'usa el canal `Alpha`, que indica transparència.

Una vegada conegudes aquestes funcions, s'elabora un *script* que siga capaç de detectar a les persones d'una imatge i generar diversos arxius PNG amb les seues siluetes amb les vores suavitzades sobre un fons transparent. El codi s'adjunta a la **Figura 35**. El seu resultat (el qual és òptim per a allò que s'està buscant) es pot veure a la **Figura 36**.

```
% Es carrega la CNN que detecta les persones (només s'inicialitza la primera vegada)

    load("maskrcnn_object_person_car_v2.mat");

% Es llegeix la imatge que conté persones

    imTest = imread("3.jpg");

% Es fa la detecció de persones

    [masks,labels,scores,boxes] = segmentObjects(net,imTest,Threshold=0.8); % Llindar baix

% Visualització de la imatge amb els individus detectats

    overlayedImage = insertObjectMask(imTest,masks);
    imshow(overlayedImage)

% Es calcula el nombre de màscares o persones detectades

    tam_mask = size(masks);
    if length(tam_mask) == 2
        num_masks = 1;
    else
        num_masks = tam_mask(3);
    end

% S'aplica cada màscara a la imatge original per a visualitzar a les persones sense fons

    for ii = 1:num_masks
        imTestmasked = uint8(double(imTest).*masks(:,:,ii));
        figure
        imshow(imTestmasked)

% Se suavitza la màscara amb un filtre gaussià de sigma = 10 per a evitar l'efecte de retall

    mascara_gauss = imgaussfilt(double(masks(:,:,ii)),10);

% Es retalla cada bbox que conté a les persones

    imTest_crop = imcrop(imTest,boxes(ii,:));
    mascara_gauss_crop = imcrop(mascara_gauss,boxes(ii,:));

% Es guarda cadascuna en un PNG amb fons transparent

    imwrite(imTest_crop, ['person_' num2str(ii) '.png'], 'Alpha', mascara_gauss_crop);

    end
```

Figura 35. *Codi de prova d'identificació de persones.*

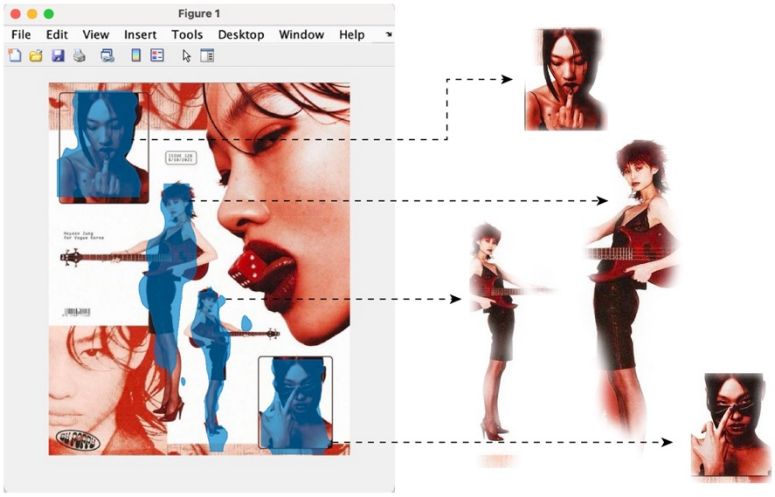


Figura 36. *Resultat efectiu per a la detecció de persones i exportació en fons transparent.*

4.3. Esbós de l'aplicació

Amb les proves fetes, s'elabora el següent esbós de l'aplicació, on s'han tingut en compte les possibilitats d'interfície que ofereix MATLAB App Designer, que és el seu entorn de desenvolupament interactiu per a dissenyar i programar aplicacions.

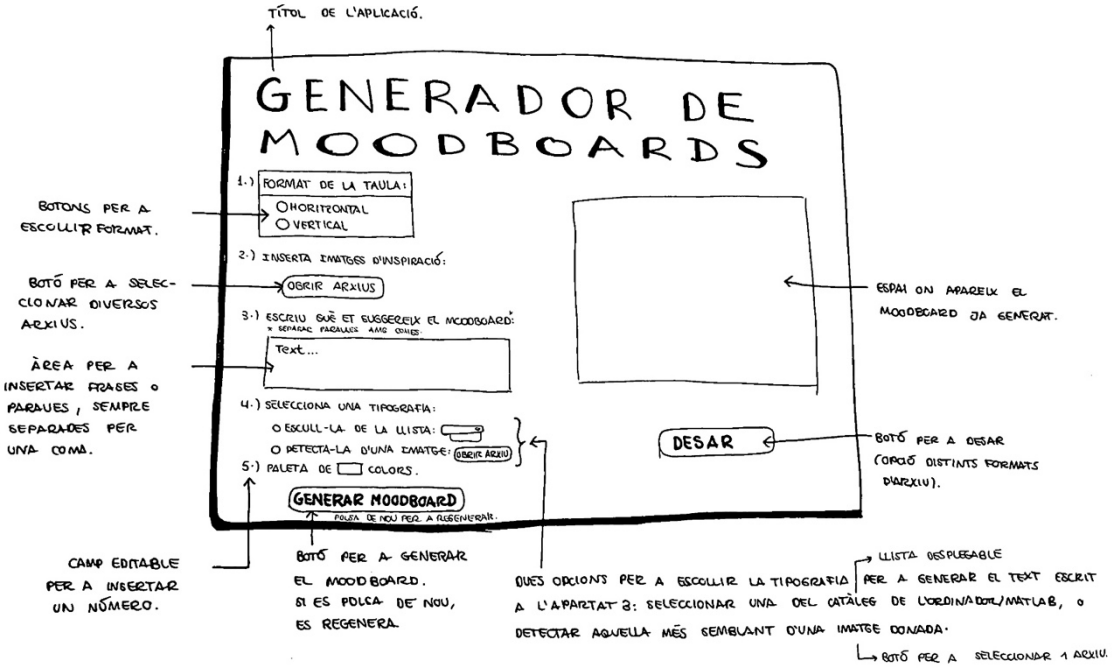


Figura 37. *Esbós del generador de moodboards.*

5. CAS PRÀCTIC: IMPLEMENTACIÓ DEL GENERADOR DE MOODBOARDS

El primer pas abans d'escriure el codi és la creació del fitxer de l'aplicació, que consta d'una vista de disseny i una vista de codi. A la vista de disseny, s'inclouen els elements plantejats a l'esbós de la **Figura 37**, donant-li un aspecte inicial el més semblant possible, com s'aprecia a la **Figura 38**. Per a poder programar cadascun d'ells, aquests elements seran declarats com a *callbacks*.

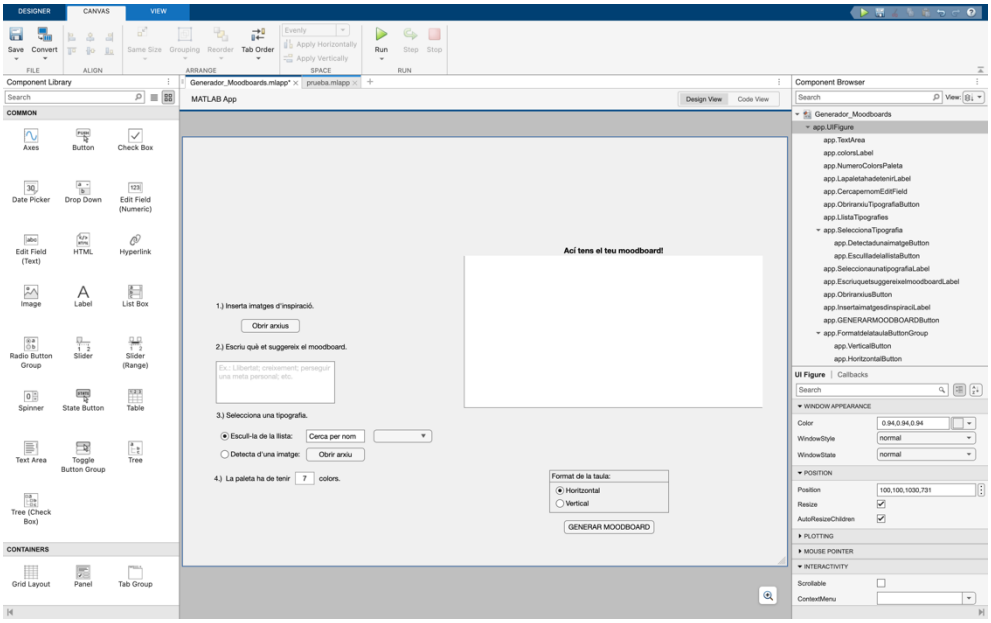
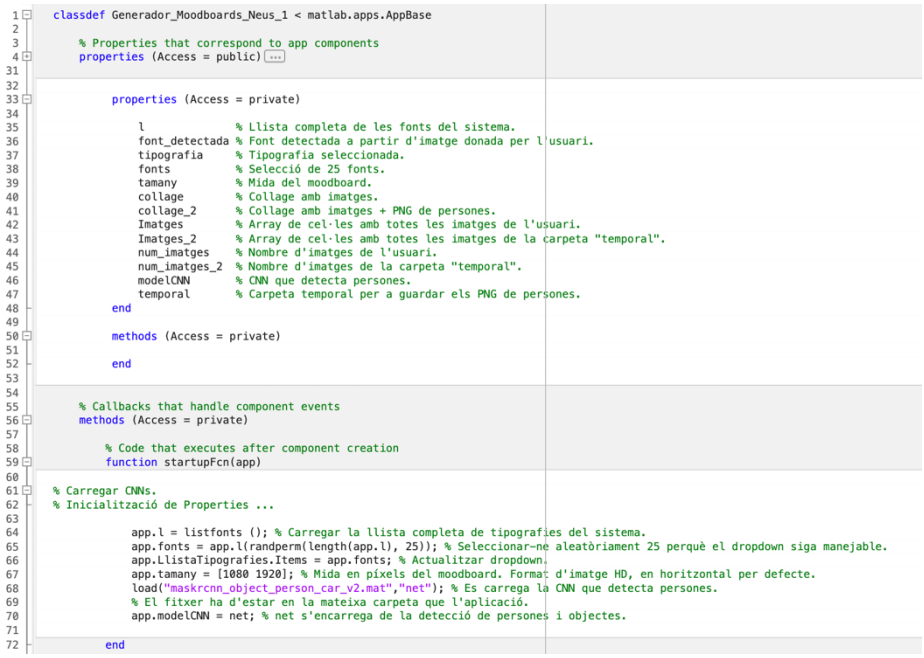


Figura 38. *Aspecte inicial de la interfície del generador de moodboards.*

5.1. Programació del codi

Seleccionant la vista del codi, es poden començar a implementar totes les funcions. A continuació, es comentarà a grans trets i per ordre de les línies del codi, què realitza cada secció. Per a aprofundir, llegiu els comentaris en color verd continguts a les figures i vegeu l'**Annex II**, que conté una taula amb l'explicació de cada comando. Convé mirar de nou el codi, ja que s'han fet petites modificacions respecte a les proves anteriors.

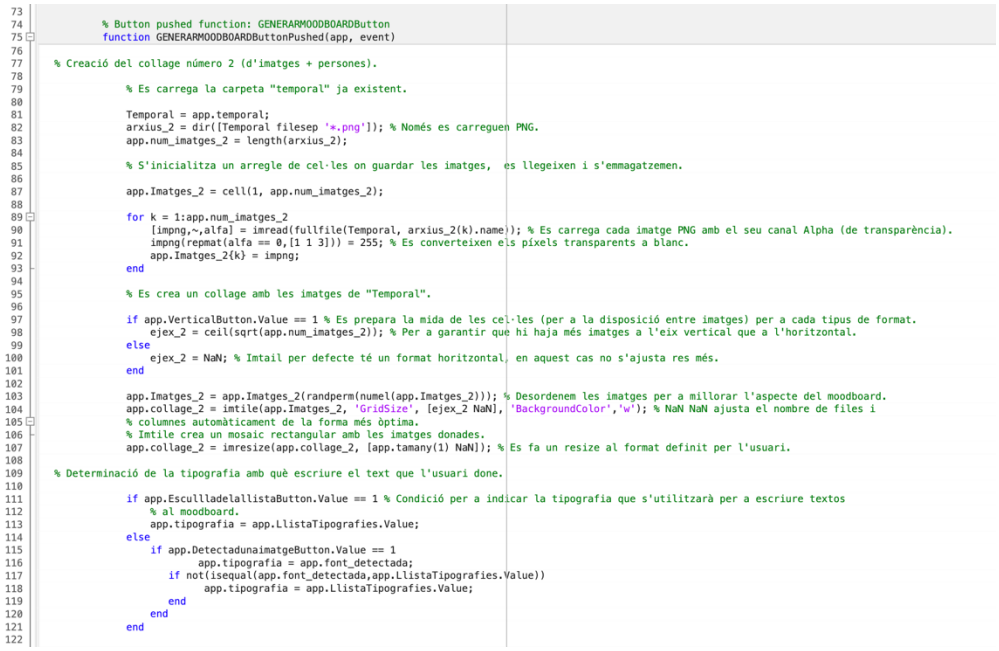
Inicialment, s'hi troben les `properties`, que són variables que emmagatzemen dades importants per a l'aplicació, com ara els components de la interfície d'usuari i dades internes com la llista de fonts i el model de xarxa neuronal (CNN). La funció `startupFcn` s'executa automàticament en crear la instància de l'aplicació i s'encarrega d'inicialitzar aquestes propietats una única vegada en obrir l'App. Per exemple, carrega la llista de fonts del sistema, selecciona aleatòriament 25 fonts per al menú desplegable, defineix la mida del *moodboard* i carrega el model de CNN per a detectar persones.



```
classdef Generador_Moodboards_Neus_1 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public) ...

        properties (Access = private)

            l              % Llista completa de les fonts del sistema.
            font_detectada % Font detectada a partir d'imatge donada per l'usuari.
            tipografia     % Tipografia seleccionada.
            fonts          % Selecció de 25 fonts.
            tamany         % Mida del moodboard.
            collage        % Collage amb imatges.
            collage_2      % Collage amb imatges + PNG de persones.
            Imatges        % Array de cel·les amb totes les imatges de l'usuari.
            Imatges_2      % Array de cel·les amb totes les imatges de la carpeta "temporal".
            num_imatges    % Nombre d'imatges de l'usuari.
            num_imatges_2  % Nombre d'imatges de la carpeta "temporal".
            modelCNN       % CNN que detecta persones.
            temporal       % Carpeta temporal per a guardar els PNG de persones.
        end

        methods (Access = private)

        end


    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)

% Carregar CNNs.
% Inicialització de Properties ...

            app.l = listfonts (); % Carregar la llista completa de tipografies del sistema.
            app.fonts = app.l(randperm(length(app.l), 25)); % Seleccionar-ne aleatòriament 25 perquè el dropdown siga manejable.
            app.LlistaTipografies.Items = app.fonts; % Actualitzar dropdown.
            app.tamany = [1080 1920]; % Mida en píxels del moodboard. Format d'imatge HD, en horitzontal per defecte.
            load("maskrcnn_object_person_car_v2.mat","net"); % Es carrega la CNN que detecta persones.
            % El fitxer ha d'estar en la mateixa carpeta que l'aplicació.
            app.modelCNN = net; % net s'encarrega de la detecció de persones i objectes.

        end
```

Figura 39. *Captura 1 del codi del generador de moodboards.*

Primer, la funció `GENERARMOODBOARDButtonPushed` s'executa quan l'usuari prem el botó "GENERAR MOODBOARD". Aquesta funció crea el *collage* final amb les imatges de la carpeta temporal, que conté tant les imatges originals sense persones com els PNG transparents amb persones retallades (que s'han processat a la funció `ObrirarxiusButtonPushed` que s'explica més endavant). El *collage* es crea utilitzant `imtile`, ajustant el nombre de files i columnes segons el format escollit per a una disposició òptima. El resultat es redimensiona a la mida especificada i es mostra a la gràfica de la interfície. A continuació, la funció determina la tipografia que s'utilitzarà per a escriure el text al *moodboard*, basant-se en si l'usuari ha seleccionat una tipografia de la llista o vol que es detecte a partir d'una imatge donada.



```
        % Button pushed function: GENERARMOODBOARDButton
        function GENERARMOODBOARDButtonPushed(app, event)

% Creació del collage número 2 (d'imatges + persones).

            % Es carrega la carpeta "temporal" ja existent.

            Temporal = app.temporal;
            arxius_2 = dir([Temporal filesep '*.png']); % Només es carreguen PNG.
            app.num_imatges_2 = length(arxius_2);

            % S'inicialitza un arregle de cel·les on guardar les imatges,  es llegeixen i s'emmagatzemen.

            app.Imatges_2 = cell(1, app.num_imatges_2);

            for k = 1:app.num_imatges_2
                [impng,~,alfa] = imread(fullfile(Temporal, arxius_2(k).name)); % Es carrega cada imatge PNG amb el seu canal Alpha (de transparència).
                impng(repmat(alfa == 0,[1 1 3])) = 255; % Es converteixen els píxels transparents a blanc.
                app.Imatges_2{k} = impng;
            end

            % Es crea un collage amb les imatges de "Temporal".

            if app.VerticalButton.Value == 1 % Es prepara la mida de les cel·les (per a la disposició entre imatges) per a cada tipus de format.
                ejex_2 = ceil(sqrt(app.num_imatges_2)); % Per a garantir que hi haja més imatges a l'eix vertical que a l'horitzontal.
            else
                ejex_2 = NaN; % Imtail per defecte té un format horitzontal, en aquest cas no s'ajusta res més.
            end

            app.Imatges_2 = app.Imatges_2(randperm(numel(app.Imatges_2))); % Desordenem les imatges per a millorar l'aspecte del moodboard.
            app.collage_2 = imtile(app.Imatges_2, 'GridSize', [ejex_2 NaN], 'BackgroundColor','w'); % NaN NaN ajusta el nombre de files i
            % columnes automàticament de la forma més òptima.
            % Imtile crea un mosaic rectangular amb les imatges donades.
            app.collage_2 = imresize(app.collage_2, [app.tamany(1) NaN]); % Es fa un resize al format definit per l'usuari.

% Determinació de la tipografia amb què escriure el text que l'usuari done.

            if app.EsculladelallistaButton.Value == 1 % Condició per a indicar la tipografia que s'utilitzarà per a escriure textos
                % al moodboard.
                app.tipografia = app.LlistaTipografies.Value;
            else
                if app.DetectadunaimatgeButton.Value == 1
                      app.tipografia = app.font_detectada;
                   if not(isequal(app.font_detectada,app.LlistaTipografies.Value))
                       app.tipografia = app.LlistaTipografies.Value;
                   end
                end
            end
```

Figura 40. *Captura 2 del codi del generador de moodboards.*

La **Figura 41** mostra el codi de la generació de paletes de color (explicat al punt 4.2.1), al qual se li apliquen modificacions menors, com el fet que no s'incloga el color blanc a les paletes generades, ja que és el color de fons del *collage*.

```
% Generació de la paleta de color.

            app.collage = double(app.collage); % Es treballa amb les imatges del primer collage creat en RGB.
            collage_c = app.collage;
            collage_c = reshape(collage_c,[size(collage_c,1)*size(collage_c,2) 3]); % Reshape per a poder treballar amb K-Means,
            % recomposant la imatge en una sola línia.

            KK = app.NumeroColorsPaleta.Value; % Número de clusters = colors que l'usuari vol en la paleta (entre 3 i 7).
            [cluster_idx] = kmeans(collage_c,KK); % Càlcul dels clústers amb K-Means.
            stats = regionprops ("Table", cluster_idx, "Area"); % Mesurar l'àrea de cada clúster.
            vector_arees = stats.Area; % Guardar les àrees en un vector.
            vector_ordenat = sort(vector_arees, 'descend'); % Ordenar les àrees de major a menor.

            RGBmitja = zeros(KK,3); % Es crea un vector de la mida dels colors de la paleta, on es guarda el valor mitjà de cada clúster.
            paleta = zeros(64,64*KK,3); % Es crea una imatge amb KK forats per a ubicar la paleta de colors.

            for ii=1:KK % Bucle per a la generació dels KK colors de la paleta.
                area = vector_ordenat(ii); % Es busquen les àrees més grans del vector d'una en una.
                idx_area = find([stats.Area] == area); % Es crea un índex que es correspon amb l'àrea més gran.
                RGBmitja(ii,:) = median(collage_c(ismember(cluster_idx,idx_area),:)); % Càlcul del valor mitjà RGB per als píxels de l'àrea.
                if not(isequal(RGBmitja(ii,:),[255 255 255])) % S'exclou el color blanc de la paleta, ja que sempre l'agafa al ser el fons
                    % del mateix color [255 255 255].
                    paleta(:,(1+64*(ii-1):64+64*(ii-1)),1) = RGBmitja(ii,1);  % Es dibuixa un quadrat de 64x64 dins de la paleta creada amb
                    % el color RGB mitjà corresponent a l'àrea.
                    paleta(:,(1+64*(ii-1):64+64*(ii-1)),2) = RGBmitja(ii,2);
                    paleta(:,(1+64*(ii-1):64+64*(ii-1)),3) = RGBmitja(ii,3);
                end
            end

            % Representació de la paleta generada.

            tam_collage = size(app.collage_2); % S'extrau la mida del collage.

            if app.HoritzontalButton.Value == 1 % Si el format del moodboard és horitzontal, es mostra la paleta en vertical.
                paleta = imrotate(paleta,90); % es rota la paleta 90° perquè quede a la dreta.
                paleta = imresize(paleta,[tam_collage(1) NaN]); % Es redimensiona la imatge "paleta" amb les files del collage i les columnes
                % que Matlab considere per a mantindre la relació d'aspecte de la imatge.
                tam_paleta = size(paleta); % S'extrau la mida de la paleta.
                moodboard = ones([tam_collage(1) tam_collage(2) + tam_paleta(2) 3])*255; % Definició de la matriu "moodboard" (files, columnes
                % totals i capes).

                moodboard(:,1:tam_collage(2),:) = app.collage_2; % Es mostra el collage a l'espai de l'esquerra.
                moodboard(:,(tam_collage(2)+1):(tam_collage(2) + tam_paleta(2)),:) = paleta; % Es mostra la paleta a l'espai de la dreta.

            else  % Si el format del moodboard és vertical, es mostra la paleta en horitzontal.

                paleta = imresize(paleta,[NaN tam_collage(2)]); % Es redimensiona la imatge "paleta" amb les files que Matlab considere per a
                % mantindre la relació d'aspecte de la imatge i les columnes del collage.
                tam_paleta = size(paleta); % S'extrau la mida de la paleta.
                moodboard = ones([tam_collage(1) + tam_paleta(1) tam_collage(2) 3]); % Definició de la matriu "moodboard" (files, columnes
                % totals i capes).

                moodboard(1:tam_collage(1),:,:) = app.collage_2; % Es mostra el collage a l'espai de dalt.
                moodboard((tam_collage(1)+1):(tam_collage(1) + tam_paleta(1)),:,:) = paleta; % Es mostra la paleta a l'espai de baix.

            end

            moodboard = uint8(moodboard); % Convertir a senxcer sense signe per a poder mostrar la imatge sense problemes.
            % Per si un cas.
            cla(app.Grafica);
            app.Grafica.Visible = "on";
            axis(app.Grafica,'off')

            imshow(uint8(moodboard),'Parent',app.Grafica); % Es mostra el collage amb la paleta.
```

Figura 41. *Captura 3 del codi del generador de moodboards.*

La funció `GENERARMOODBOARDButtonPushed` acaba amb la generació i disposició aleatòria del text escrit a la interfície de l'App amb la tipografia que corresponga.

```
% Obtenció i representació del contingut de l'àrea de text.

            paraules = split(app.TextArea.Value, ';'); % Es fa una partició de la matriu de l'àrea de text per a obtenir cadascuna de les
            % paraules.
            paraula_max = max(strlength(paraules)); % Càlcul de la longitud de la paraula més llarga.
            x = randperm(tam_collage(2)-(paraula_max+2)*24,numel(paraules)); % Càlcul de les posicions aleatòries del text guardant un marge
            % amb el text més llarg.
            y = randperm(tam_collage(1)-24,numel(paraules));

            hold(app.Grafica, 'on'); % Mantenir els elements existents a la gràfica.

            for i = 1:numel(paraules) % Es recorren totes les paraules i es mostren d'una en una a la Gràfica.
                text(app.Grafica, x(i), y(i), paraules{i}, 'Color', 'k','BackgroundColor','w', 'FontSize', 24, 'FontUnits', 'pixels', 'FontName', app.tipografia);
                % S'usa la tipografia més semblant o la escollida.
            end

            hold(app.Grafica, 'off'); % S'allibera la gràfica per a possibles operacions futures.

            app.Grafica.Title.String = 'Ací tens el teu moodboard!';  % Es torna a escriure per si un cas s'ha esborrat durant el procés.
            app.Grafica.Title.Color = [0.99 0.96 0.79];

        end
```

Figura 42. *Captura 4 del codi del generador de moodboards.*

El següent *callback* s'anomena `ObrirarxiusButtonPushed`. S'executa quan l'usuari prem el botó "Obrir arxius". Aquesta funció permet seleccionar una carpeta mitjançant un diàleg de selecció (`uigetdir`), mostrant un missatge d'advertència si no se selecciona cap arxiu. En cas contrari, es carreguen els noms dels fitxers d'imatge (el format dels quals s'ha escollitl a criteri propi) de la carpeta seleccionada. Si no es troben imatges, es mostra un missatge d'error. Si es troben imatges, es crea un array (o arregle) de cel·les per a emmagatzemar-les, i cada imatge es llegeix i es guarda en aquest array.

```
        % Button pushed function: ObrirarxiusButton
        function ObrirarxiusButtonPushed(app, event)

% Obertura d'arxius.

            % Selecció de la carpeta.

            Carpeta = uigetdir();

            % En cas de no seleccionar cap carpeta, es mostra el següent missatge: "Cap carpeta seleccionada".

            if isequal(Carpeta, 0)
                warndlg('Cap carpeta seleccionada');
                return;
            end

            % Es carreguen els noms dels arxius amb les imatges de la carpeta

            arxius = [...
                dir([Carpeta filesep '*.jpg']);
                dir([Carpeta filesep '*.jpeg']);
                dir([Carpeta filesep '*.png']);
                dir([Carpeta filesep '*.bmp']);
                dir([Carpeta filesep '*.tif']);
                dir([Carpeta filesep '*.tiff']);
                ];

            % Es mostra un missatge d'error en cas de no haver-hi imatges a la carpeta.

            app.num_imatges = length(arxius);

            if app.num_imatges == 0
                warndlg('No s`han trobat imatges a la carpeta seleccionada');
                return;
            end

            % S'inicialitza un arregle de cel·les on guardar les imatges,  es llegeixen i s'emmagatzemen.

            app.Imatges = cell(1, app.num_imatges);

            for k = 1:app.num_imatges
                app.Imatges{k} = imread(fullfile(Carpeta, arxius(k).name)); % Sols s'obrin arxius de tipus imatge.
            end
```

Figura 43. *Captura 5 del codi del generador de moodboards.*

Dintre de la mateixa funció, es decideix incloure la detecció de persones. Emprant la CNN, si es troben individus, s'aplica una màscara a cada silueta detectada per a eliminar el fons i suavitzar les vores amb un filtre gaussià. La silueta es guardarà com un fitxer PNG amb fons transparent a la carpeta temporal creada. Lògicament, les imatges originals d'on s'han extret les persones no es tindran en compte per al *collage* final.

```
% Detecció de persones a les imatges per a eliminar-ne el fons i incloure els PNG extrets a la carpeta d'imatges donada.

            app.temporal = [Carpeta filesep 'temporal']; % Es crea una carpeta temporal dintre de 'Carpeta'.
            [status,~,~] = mkdir(app.temporal); % mkdir crea una nova carpeta.
            if status == 0 % Si la creació de la carpeta falla.
                warndlg('No s`ha pogut crear la carpeta temporal');
                return;
            end
            delete([app.temporal filesep '*.png']);  % S'esborren tots els PNG que pogués haver abans a la carpeta "temporal".

            % Es busquen les imatges que contenen persones.

            for k = 1:app.num_imatges
                [masks, labels, ~, boxes] = segmentObjects(app.modelCNN,app.Imatges{k},Threshold = 0.8); % Es fa la
                % detecció de persones. Llindar baix.
            esPersona = zeros (size(labels)); % Nou vector de la mida de labels.
            esPersona (labels == 'person') = 1; % Si es tracta d'una persona, la posició del vector serà 1.
                if any(esPersona) % Si hi ha persones.
                    % Es calcula el nombre de màscares o persones detectades.
                    tam_mask = size(masks);
                    if length(tam_mask) == 2
                        num_masks = 1;
                    else
                        num_masks = tam_mask(3);
                    end
                    % S'aplica cada màscara a la imatge original per a eliminar-ne el fons.
                    for ii = 1:num_masks
                       if esPersona(ii) == 1 % Processar només les màscares de persones.
                       mascara_gauss = imgaussfilt(double(masks(:,:,ii)),10); % Se suavitza la màscara amb un filtre gaussià de
                       % sigma = 10 per a evitar l'efecte de retall.
                       imatgeOriginal = app.Imatges{k};
                       imatge_crop = imcrop(imatgeOriginal,boxes(ii,:)); % Es retalla cada bbox que conté a les persones.
                       mascara_gauss_crop = imcrop(mascara_gauss,boxes(ii,:));
                       imwrite(imatge_crop, [app.temporal filesep 'person_' num2str(k) '_' num2str(ii) '.png'], 'Alpha', mascara_gauss_crop);  % Es
                       % guarda cadascuna en un PNG amb fons transparent.
                       end
                    end
                else
                    imwrite(app.Imatges{k}, [app.temporal filesep 'no_person_' num2str(k)  '.png']);  % Guardar la imatge completa sense persones
                    % en la carpeta "temporal".
                end
            end
```

Figura 44. *Captura 6 del codi del generador de moodboards.*

La funció acaba mostrant el *collage* provisional a l'espai de la dreta de la interfície.

```
% Es mostra un collage amb les imatges de "Carpeta" en app.Grafica.

            if app.VerticalButton.Value == 1 % Es prepara la mida de les cel·les (per a la disposició entre imatges) per a cada tipus de format.
                ejex = ceil(sqrt(app.num_imatges)); % Per a garantir que hi haja més imatges a l'eix vertical que a l'horitzontal.
            else
                ejex = NaN; % Imtail per defecte té un format horitzontal, en aquest cas no s'ajusta res més.
            end

            app.collage = imtile(app.Imatges, 'GridSize', [ejex NaN], 'BackgroundColor','w'); % NaN NaN ajusta el nombre de files i
            % columnes automàticament de la forma més òptima.
            % Imtile crea un mosaic rectangular amb les imatges donades.
            app.collage = imresize(app.collage, [app.tamany(1) NaN]); % Es fa un resize al format definit per l'usuari.

            % Per si un cas.
            cla(app.Grafica);
            app.Grafica.Visible = "on";
            axis(app.Grafica,'off')

            imshow(app.collage,'Parent',app.Grafica) % Es mostra el collage provisional en app.Grafica.
            app.Grafica.Title.String='Collage';
            app.Grafica.Title.Color = [0.99 0.96 0.79];

        end
```

Figura 45. *Captura 7 del codi del generador de moodboards.*

El següent fragment de codi gestiona la selecció de tipografia en l'aplicació. La funció `SeleccionaTipografiaSelectionChanged` s'activa quan l'usuari prem els botons `DetectadunaimatgeButton` o `EsculladelallistaButton`. Depenent de la selecció, el botó `ObrirarxiuTipografiaButton` es mostra o s'oculta. D'altra banda, la funció `LlistaTipografiesValueChanged` s'activa quan es canvia la selecció de la llista de tipografies, actualitzant la tipografia seleccionada amb el valor escollit.

```
        % Selection changed function: SeleccionaTipografia
        function SeleccionaTipografiaSelectionChanged(app, event)

% Aquest callback s'activa per a dos casos: si l'usuari prem el botó app.DetectadunaimatgeButton o si prem el de app.EsculladelallistaButton

            switch app.SeleccionaTipografia.SelectedObject.Text
                case "Detecta d'una imatge:"
                    app.ObrirarxiuTipografiaButton.Visible = "on";
                case "Escull-la de la llista:"
                    app.ObrirarxiuTipografiaButton.Visible = "off";
            end

        end

        % Value changed function: LlistaTipografies
        function LlistaTipografiesValueChanged(app, event)

% Actualització de la tipografia seleccionada

             app.tipografia = app.LlistaTipografies.Value;

        end
```

Figura 46. *Captura 8 del codi del generador de moodboards.*

La **Figura 47** mostra el codi que gestiona la funció del botó `ObrirarxiuTipografiaButton`. Quan es prem aquest botó, l'usuari selecciona una imatge d'un fitxer. Si no es selecciona cap fitxer, es mostra un avís. Després, se segueix pràcticament el mateix codi explicat a la **Figura 33**. Finalment, s'actualitza la llista desplegable de tipografies amb la tipografia detectada per l'OCR.

```
        % Button pushed function: ObrirarxiuTipografiaButton
        function ObrirarxiuTipografiaButtonPushed(app, event)


            % Primer, s'inicia la funció per a l'obertura d'un arxiu de diferents tipus possibles.

            [filename, pathname] = uigetfile({'*.jpg;*.tif;*.png;*.gif','All Image Files'; '*.*','All Files'},'Selecciona una imatge d`un text...');

            if isequal(filename,0)
                    warndlg('Has de seleccionar una imatge amb text');
                    return
            end

            imatge = imread(strcat(pathname,filename));

            % Es detecta la posició del text en la imatge.

            bboxes = detectTextCRAFT(imatge); % Es la 'Region Of Interest' (roi).

            % Es crida a l'OCR i se li dona la ubicaió de les paraules.

            ocrResults = ocr(imatge,bboxes,'TextLayout', 'block');

            % S'extrau el text trobat.

            words = ocrResults(:).TextLines; % Línies de text reconegudes per l'OCR.
            newbbox = ocrResults(:).TextLineBoundingBoxes; % Coordenades de les bounding boxes.

            % S'ajusta la imatge donada a la mida de la bounding box.

            textimgcrop = imcrop(imatge,newbbox);

            % Es passa la imatge retallada a escala de grisos i s'inverteix.

            Tipografia = 255-im2gray(textimgcrop); % S'inverteix la imatge (fons negre i figura en blanc).
            tamtipo = size(Tipografia); % S'agafa la mida del text de la imatge donada.

            maxcoef = 0; % Declaració variable per a emmagatzemar el coeficient de major valor.
            maxindex = 1;

            app.Grafica.Title.String='';  % Important perquè no es veja.

            app.Grafica.Visible = "off";

            % Bucle per a comparar la imatge donada amb totes les tipografies de la llista reduïda.

            for ii = 1:size(app.fonts) % "fonts" és la llista reduïda a només 25 tipografies
                cla(app.Grafica); % S'esborra l'anterior figura
                text(app.Grafica, 0.1, 0.5, words, 'Color', 'k','BackgroundColor','w','FontSize', 36, 'FontName', app.fonts{ii}) % Escriure el
                % text detectat per l'OCR amb la tipografia ii.
                exportgraphics(app.Grafica,'Prueba.png') % Exportar la figura en una imatge .PNG.
                Prueba = 255-im2gray(imread("Prueba.png")); % Importar la imatge en negatiu.
                Prueba = imresize(Prueba,tamtipo); % Ajustar a la mida de la imatge donada inicialment per l'usuari.
                coef = corr2(Prueba,Tipografia); % Càlcul del coeficient de correlació d'ambdues imatges.
                if coef > maxcoef
                    maxcoef = coef; % Actualització del coeficient màxim.
                    maxindex = ii; % Actualització de la posició de la llista del coeficient màxim.
                end

            end

            cla(app.Grafica);
            app.Grafica.Visible = "on";
            axis(app.Grafica,'off')

            app.font_detectada = app.fonts{maxindex};

            % Actualització de la tipografia seleccionada en el dropdown.

            app.LlistaTipografies.Items = app.fonts; % Per si un cas s'ha canviat abans.
            app.LlistaTipografies.Value = app.font_detectada;

        end
```

Figura 47. *Captura 9 del codi del generador de moodboards.*

Per acabar, el *callback* `CercapernomEditField` captura el valor de text que l'usuari li puga donar, filtrant la llista de fonts disponibles per aquelles que contenen el text introduït, ignorant majúscules i minúscules, i s'actualitza la llista de tipografies mostrades (`LlistaTipografies`). Quan es prem el botó "Desar", es mostra una caixa de diàleg perquè l'usuari seleccione el nom i la ubicació per desar un fitxer PNG del *moodboard*, i s'exporta la gràfica amb el nom i la ubicació seleccionats.

```
        % Value changed function: CercapernomEditField
        function CercapernomEditFieldValueChanged(app, event)

            % Obtenir el valor del EditField

            Cercarpernom = app.CercapernomEditField.Value;

            % Filtrar les fonts en funció del text escrit en el EditField

            llista_filtrada= app.l(contains(app.l, Cercarpernom, 'IgnoreCase', true));

            % Actualizar els elements de la llista de fonts segons el filtre

            % app.LlistaTipografies.Items = app.fonts;

            app.LlistaTipografies.Items = llista_filtrada;

        end

        % Button pushed function: DesarButton
        function DesarButtonPushed(app, event)

        [file, path] = uiputfile('*.png', 'Desar moodboard com'); % Caixa de diàleg per a insertar nom i ubicació.
        nomArxiu = fullfile (path, file);
        app.Grafica.Title.Color = [0.15 0.15 0.15];
        exportgraphics(app.Grafica, nomArxiu);

        end
    end
```

Figura 48. *Captura 10 del codi del generador de moodboards.*

5.2. Resultats

Després de provar el funcionament de l'aplicació, es canvia el disseny de la interfície per a fer la visualment més atractiva. El resultat és el següent.

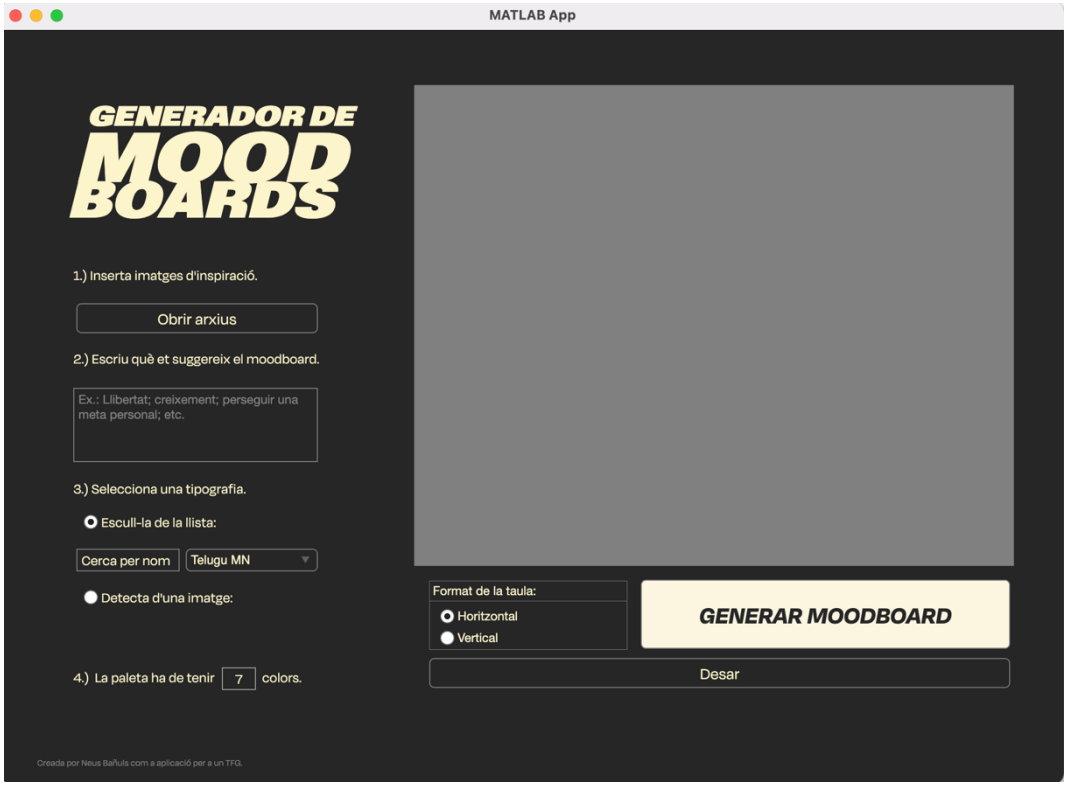


Figura 49. *Interfície final del generador de moodboards.*

Si es compila el codi, el resultat és el següent: l'App demana que s'adjunte una carpeta. Aquesta, ha d'estar prèviament preparada amb imatges (almenys, una d'elles ha de contenir gent perquè s'aprecie la detecció de persones). Després, mentre es processen les imatges mitjançant la CNN, es poden anar emplenant la resta d'apartats, on es demana que s'escriga text d'inspiració (separat per punts i comes), es trie de quina forma es vol escollir la tipografia amb què s'escriurà el text, s'introduïsca un nombre de colors per a la paleta i es trie el format del *moodboard*.

Quan apareix un collage provisional a l'espai de la dreta (com a la **Figura 50**), significa que la CNN ha acabat. En aquest moment, s'activa el botó "Obrir arxiu" que permet adjuntar una imatge per a detectar una tipografia (si és que es vol emprar i no s'ha emprat fins al moment), i ja es pot prémer el botó "GENERAR MOODBOARD".

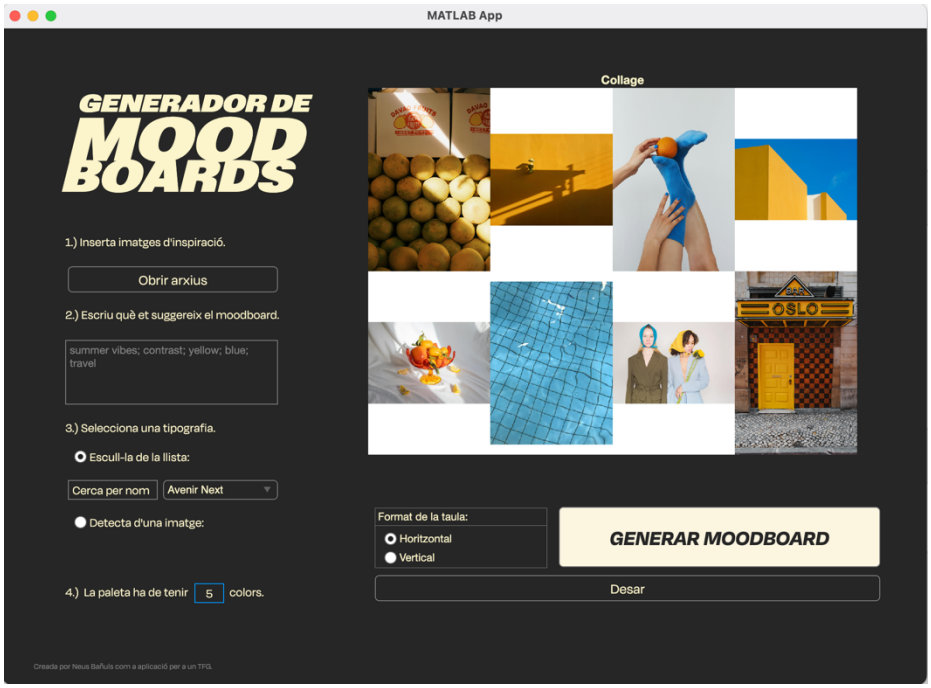


Figura 50. *Interfície final del generador de moodboards després d'obrir la carpeta d'arxius.*

Amb "GENERAR MOODBOARD" es genera el producte final, que es pot regenerar tantes vegades com es desitge prement-lo de nou. En aquest cas, convé regenerar-lo fins que el text es dispose en ubicacions que no tapen altres elements importants.

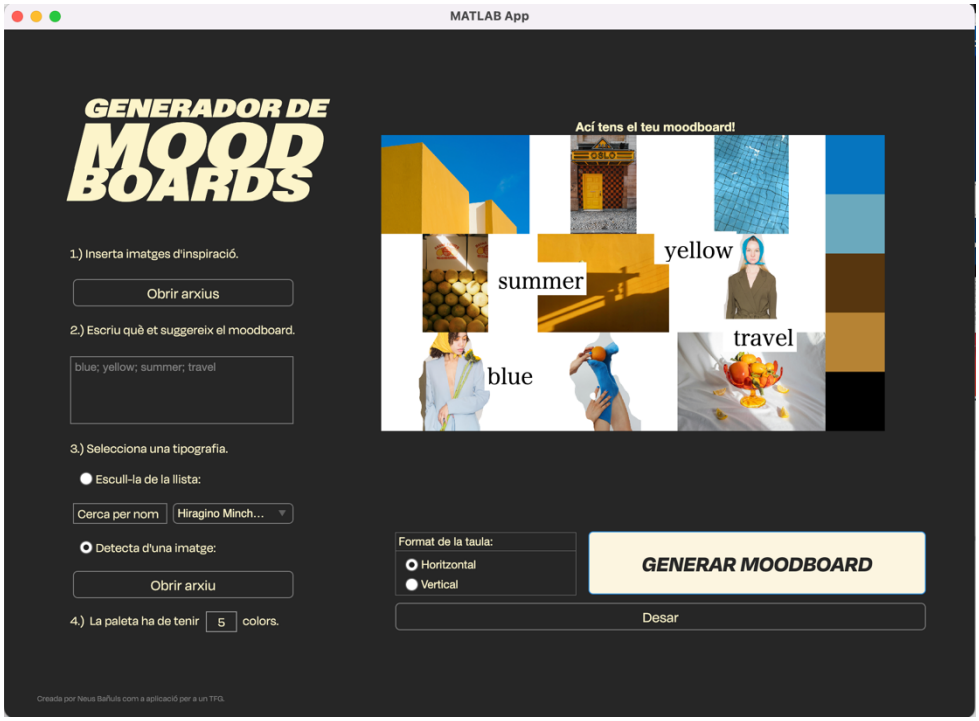


Figura 51. *Interfície final després de generar el moodboard.*

Finalment, quan s'obté el millor resultat, es prem el botó "Desar" per a guardar el *moodboard* com un arxiu.

Figura 52. *Moodboard final generat mitjançant l'aplicació.*

Originalment, la idea era que les siluetes de les persones se superposaren a les altres imatges i que el text es disposara de forma que no ocultara cap element; també hagués sigut ideal que es pogués retallar la silueta de qualsevol objecte que tinguera les vores ben definides (que fora fàcilment separable del seu fons), però els recursos disponibles a MATLAB, el temps i els coneixements limitats han suposat que la idea inicial canvie una mica de rumb. No obstant això, el resultat és similar al plantejat inicialment, ja que, d'una forma o altra, es compleix amb tots els objectius que s'han proposat. L'important és que s'ha obtingut una base sobre la qual es poden fer avanços.

5.3. Possibles usos de l'aplicació

Veient els resultats obtinguts, l'App podria integrar-se com a *plug-in* dels programes Adobe Illustrator i Adobe Photoshop o, fins i tot, podria ser una de les ferramentes que contenen aquests programes. Només caldria donar-li els arxius necessaris i emplenar els camps requerits per a l'obtenció de la taula d'inspiració.

D'altra banda, si el generador de *moodboards* es perfeccionara i s'inclogueren més funcions, podria ser una aplicació independent que oferira uns resultats ja més polits. Fins i tot, en un cas hipotètic, podria implementar-se la IA perquè no segués necessari adjuntar arxius i, només amb descriure de què es vol el *moodboard*, l'App el creara des de zero.

6. CONCLUSIONS

En aquest TFG, s'ha desenvolupat una aplicació en MATLAB per a la creació automàtica de *moodboards*, integrant tècniques avançades de tractament digital d'imatges (TDI) i *deep learning*.

L'objectiu principal era crear una eina que permetés als dissenyadors plasmar visualment les seues idees mitjançant *collages* generats a partir d'unes dades proporcionades. Durant el procés, s'ha aconseguit tant aquest objectiu com els diversos específics, que són l'estudi de tècniques de TDI, el testatge de la detecció i reconeixement de tipografies i persones, i la integració d'un generador de paletes de color.

Pel que fa a l'aspecte acadèmic, es determina que seguir la metodologia marcada ha sigut eficaç per a assolir la meta establerta a l'inici del TFG. La investigació prèvia és indispensable per a la correcta execució de l'aplicació, on cada funció se sustenta sòlidament per un fonament teòric científic. No es pot programar sense conéixer les arrels teòriques que fonamenten cada tècnica utilitzada.

El procés metodològic seguit ha inclòs diverses etapes clau, començant per la ideació i la conceptualització del projecte, seguida per una fase de proves amb MATLAB per assegurar la viabilitat de les tècniques escollides. Aquestes proves han sigut crucials per ajustar i millorar la funcionalitat de cada mòdul de l'aplicació, assegurant-ne un rendiment òptim.

Després d'aquesta fase inicial, la implementació del codi ha estat realitzada amb rigor, seguint les millors pràctiques de programació i assegurant la robustesa i eficàcia de l'App. La integració del generador de paletes de color, desenvolupat prèviament en el curs acadèmic, ha aportat un valor afegit al projecte, demostrant la importància de reutilitzar i adaptar recursos existents per a noves aplicacions.

Finalment, la validació de l'aplicació mitjançant tests ha permés detectar i corregir possibles errors, assegurant que l'eina compleix amb els objectius establerts i ofereix una experiència d'usuari satisfactòria. Aquest procés ha sigut fonamental per a garantir que l'App no només siga teòricament sòlida, sinó també pràcticament útil i eficaç.

Una vegada finalitzat el treball, s'han observat també diversos aspectes rellevants. En primer lloc, s'ha de tenir present que el resultat obtingut és només una base que pot donar peu a l'elaboració d'una aplicació, ferramenta o *plug-in* més complex i elaborat. Sabent açò, es pot concloure que s'ha obtingut una aplicació que funciona raonablement bé, demostrant així la viabilitat de les tècniques proposades per a l'automatització dels *moodboards*. Aquest és l'aspecte que més valor li dona al TFG, ja que s'ofereix una solució innovadora per a un problema existent al camp del disseny, la qual és perfectament implementable i extrapolable a programes com Illustrator i Photoshop. A més a més, amb la creació de l'aplicació s'han pogut mesclar el món artístic amb el tecnològic, demostrant que ambdós poden anar de la mà, i posant en pràctica el Doble Grau en Enginyeria de Sistemes de Telecomunicació, So i Imatge + Comunicació Audiovisual.

En segon lloc, es determina que la creació d'un producte tecnològic com aquest és fruit d'un constant prova-error i d'una sèrie d'investigacions de les quals mai s'obté un resultat idèntic al que es planteja al principi. La idea inicial ha anat canviant fins a adoptar la seua forma final, açò demostra que, com a enginyers, s'ha de ser flexible i adaptar-se als recursos que la tecnologia ens proporciona.

El projecte ha revelat la complexitat darrere d'accions aparentment simples com detectar una tipografia o disposar imatges en forma de mosaic òptimament, i com una sola funció pot requerir moltes hores de treball. Cada línia de codi és fonamental per al bon funcionament del sistema, i qualsevol detall pot marcar la diferència entre tenir errors o no. Sens dubte, solucionar errates ha estat una constant en aquest treball.

En definitiva, en aquest TFG no només s'han assolit els objectius establerts, sinó que també s'han sabut integrar les tècniques estudiades en el desenvolupament d'aplicacions pràctiques. Així mateix, l'experiència adquirida durant el procés ha sigut molt valuosa, ja que s'ha ampliat el llenguatge de programació aprés amb MATLAB. Finalment, el més important ha sigut solucionar de forma efectiva un problema real del món artístic, donant peu a un futur desenvolupament del producte.

7. BIBLIOGRAFIA

ALORF, A. (2017): *K-Means, Mean Shift, and SLIC Clustering Algorithms: A Comparison of Performance in Color-based Skin Segmentation*, Tesi magistral, Universitat de Pittsburgh.

DALAL, N. i TRIGGS, B. (2005): *Histograms of oriented gradients for human detection*, 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego.

GONZALEZ, R. C., WOODS R. E. i EDDINS S. L. (2004): *Digital Image Processing Using MATLAB*, Nova Jersey, Prentice Hall.

MAGRO, R. (2013): *Binarización de imágenes digitales y su algoritmia como herramienta aplicada a la ilustración entomológica*, Valladolid, Butlletí de la Societat Entomològica Aragonesa (S.E.A.), nº 53.

MITA, T., KANEKO, T, i HORI, O. (2005): *Joint Haar-like features for face detection*, Tenth IEEE International Conference on Computer Vision (ICCV'05), Beijing (Xina).

MONTORO, S. (2015): *Repaso didáctico sobre Machine Learning*, Disponible en <https://lapastillaroja.net/2015/02/ml-algols/> [Consulta: 23-05-2024].

PLATERO, C. (2007): «*Capítulo 3: Procesamiento digital de imágenes*», *Apuntes de Visión Artificial*, Elx, Departament d'Electrònica, Automàtica i Informàtica Industrial de la Universitat Miguel Hernández (UMH).

PLATERO, C. (2007): «*Capítulo 5: Segmentación*», *Apuntes de Visión Artificial*, Elx, Departament d'Electrònica, Automàtica i Informàtica Industrial de la Universitat Miguel Hernández (UMH).

RAMOS, A. (2017): *Seguimiento de objetos por visión (Asset Tracking)*, Treball final de carrera, Universitat Politècnica de Madrid.

REAL ACADEMIA ESPAÑOLA (2014[23]): En *Diccionario de la lengua española*. Disponible en <https://dle.rae.es/correlación?m=form> [Consulta: 28-05-2024].

RODRÍGUEZ, B. (2008): *Segmentación de imágenes en color y textura basada en el modelo de los bordes locales (LEP)*, Treball final de carrera, Universitat de Sevilla, Disponible en <https://biblus.us.es/bibing/proyectos/abreproy/11527/> [Consulta: 19-07-2024].

SAPPA, A. i DEVY, M. (2001): *Fast range image segmentation by an edge detection strategy*, Tercera Conferència Internacional en 3-D Digital Imaging and Modeling, França, Disponible en <https://www.researchgate.net/publication/3897575_Fast_range_image_segmentation_by_an_edge_detection_strategy> [Consulta: 18-05-2024].

The MathWorks Inc. (2024): *Introducción a Deep Learning*, Natick, Massachusetts: The MathWorks Inc., Disponible en <https://es.mathworks.com/discovery/deep-learning.html> [Consulta: 22-05-2024].

VINUESA, P. (2016): *Tema 8 - Correlación: teoría y práctica*, Disponible en <https://www.ccg.unam.mx/~vinuesa/R4biosciences/docs/Tema8_correlacion.html> [Consulta: 29-05-2024].

8. PER A SABER-NE MÉS

Faster Capital (2024): *El poder de los coeficientes de correlacion cruzada en el analisis de datos*, Disponible en <https://fastercapital.com/es/contenido/El-poder-de-los-coeficientes-de-correlacion-cruzada-en-el-analisis-de-datos.html> [Consulta: 28-05-2024].

OLAYA, V. (2020): *Sistemas de Información Geográfica*, Disponible en <https://volaya.github.io/libro-sig/chapters/Imagenes.html> [Consulta: 20-05-2024].

The MathWorks Inc. (2024): *How does the xcorr fun works and what is the difference between corr and xcorr?,* Natick, Massachusetts: The MathWorks Inc., Disponible en <https://es.mathworks.com/matlabcentral/answers/1726240-how-does-the-xcorr-fun-works-and-what-is-the-difference-between-corr-and-xcorr> [Consulta: 20-05-2024].

YOUTUBE:

SignalSense (28 de juliol de 2020): *Cómo los INGENIEROS buscan a WALLY | CORRELACIÓN CRUZADA de IMÁGENES.* [Arxiu de Vídeo]. YouTube. https://youtu.be/sfsIhX2lQxw?si=NEbaX38B6SuHgzF3

DORRAN, D. [David Dorran] (24 de febrer de 2014): *Correlation Explanation with Demo.* [Arxiu de Vídeo]. YouTube. https://youtu.be/_r_fDlM0Dx0?si=mz7hbpadrweoPfP1

DORRAN, D. [David Dorran] (25 de febrer de 2014): *Normalised Correlation Explanation with Demo.* [Arxiu de Vídeo]. YouTube. https://youtu.be/ngEC3sXeUb4?si=w4ad_dow_NMK_oBH

DORRAN, D. [David Dorran] (25 d'abril de 2014): *Cross Correlation Demo using Matlabs xcorr function.* [Arxiu de Vídeo]. YouTube. https://youtu.be/RO8s1TrElEw?si=dLQ9Ye9qWsHcP_T6

SkillC (03 de juny de 2021): *HOG Intuition | Simple Explanation | Feature Descriptor & Engineering.* [Arxiu de Vídeo]. YouTube. https://youtu.be/5nZGnYPyKLU?si=NactbHijkMEmVl0l