Albert Cañellas Solé

Aplicació d’escriptori pel control de l’aforament en Python

TREBALL DE FI DE GRAU

dirigit per Hatem Abdellatif Fatahallah Ibrahim Mahmoud

Grau d'Enginyeria Informàtica i Biotecnologia

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2021

Resum.

A causa de l'actual pandèmia de la COVID-19, s'ha de controlar l'aforament en espais tancats, com botigues i cinemes. Aquest projecte es bassa en cobrir aquesta necessitat des del punt de vista informàtic, implementant una aplicació d'ordinador que permet controlar l'aforament desitjat dins d'un establiment.

L'aplicació analitzarà i detectarà, a fotograma les persones, permeten així calcular quantes persones entrarien en l'establiment i quantes en surten. Per programar l'aplicació s'utilitzarà *Python* i el *VSCode*, i per l'anàlisi i detecció de persones la llibreria *OpenCV* i una xarxa neuronal. El resultat es mostrarà per pantalla, gràcies a la implementació d'una interfície gràfica amb l'ajuda de *Tkinter*.

Resumen.

Debido a la actual pandemia del COVID-19, se debe controlar el aforo en espacios cerrados, como tiendas y cines. Este proyecto se basa en cubrir esta necesidad desde el punto de vista informático, implementando una aplicación de ordenador que permite controlar el aforo deseado dentro de un establecimiento.

La aplicación analizará y detectará, a fotograma las personas, permiten así calcular cuántas personas entrarían en el establecimiento y cuántas salen. Para programar la aplicación utilizará *Python* y el *VSCode*, y por el análisis y detección de personas la librería *OpenCV* y una red neuronal. El resultado se mostrará por pantalla, gracias a la implementación de una interfaz gráfica con la ayuda de *Tkinter*.

Abstract.

Due to the current COVID-19 pandemic, capacity must be controlled in enclosed spaces, such as shops and cinemas. This project is based on covering this need from a computer point of view, implementing a computer application that allows you to control the desired capacity within an establishment.

The application will analyze and detect, in person, to calculate how many people would enter the establishment and how many would leave. *Python* and *VSCode* will be used to program the application, and the OpenCV library and a neural network will be used to analyze and detect people. The result will be displayed on the screen, thanks to the implementation of a graphical interface with the help of *Tkinter*.

Índex

Índex de taules

Índex de figures

1 Introducció

Actualment la informàtica i la tecnologia es troben presents a tot arreu. Ajudant en la majoria de tasques fins i tot en les més petites, com estudiant del doble grau en Enginyeria Informàtica i Biotecnologia, crec que la biotecnologia es pot beneficiar de la informàtica.

En els temps actuals estem vivint una pandèmia mundial que ha ficat sota les cordes al propi sistema i provocant un canvi en la manera d'actuar a la que estàvem acostumats.

La pandèmia és deguda al virus COVID-19, és una malaltia infecciosa causada per SARS-CoV-2, un virus estretament relacionat amb el SARS-CoV, però amb una capacitat de disseminació mundial molt més alta que aquest va provocar entre els anys 2002 i 2003. El primer cas es va identificar a Wuhan, província de Hubei, Xina, el desembre de 2019. Inicialment, la malaltia es pot presentar asimptomàtica o amb pocs símptomes, o pot aparèixer febre, tos seca o amb producció d'esputs, sensació de falta d'aire, afectació de l'olfacte, dolor als músculs i cansament. També s'han notificat símptomes gastrointestinals com la diarrea. La majoria dels casos són lleus, però amb un empitjorament pot aparèixer una pneumònia, síndrome del destret respiratori, una síndrome d'alliberament de citocines, una disfunció multiorgànica i la mort. A data de 5 d'agost 2021, s'han produït aproximadament, 4.263.000 morts confirmades i 200.702.000 casos confirmats.

S'ha observat afectació a més llarg termini en els òrgans (en particular els pulmons i el cor), i hi ha un nombre significatiu de pacients que recuperats de la fase aguda de la malaltia, continuen experimentant una sèrie d'efectes en el que s'ha vingut a anomenar COVID persistent.

El SARS-CoV-2 s'estén per diversos mitjans, principalment per la saliva i altres fluids i secrecions corporals. La saliva i els fluids respiratoris poden formar petites gotes i aerosols, que poden escampar a mesura que una persona infectada respira, tus, esternuda, canta o parla. El virus també es pot propagar per contacte directe i es desconeix la freqüència amb què es propaga per fòmits (superfícies contaminades). La ruta de transmissió exacta poques vegades es demostra de manera concloent, però la infecció es produeix principalment quan les persones estan pròximes les unes a les altres durant un temps suficient, el que es coneix com a "contacte estret" [1].

La transmissió es pot limitar amb una bona higiene que inclou la rentada de mans freqüent i correcte, mantenir una distància de seguretat entre altres persones i no tocar-se la a cara (després de possible contaminació).

Com que no s'espera que una vacuna del tot eficient estigui disponible aviat, a causa de les recents mutacions del virus, una qüestió clau en la gestió de la pandèmia COVID-19 és intentar disminuir el pic de l'epidèmia i aplanar la corba epidèmica, a través de diverses mesures que pretenen reduir la taxa de noves infeccions. La disminució del nombre reproductiu bàsic o també conegut com a "R_0", ajuda a disminuir el risc de col·lapsar els serveis sanitaris, permeten un millor tractament dels casos greus actuals i proporcionant més temps perquè es desenvolupi un tractament. Amb una "R_0" inferior a 1 la infecció tendirà a desaparèixer i si és superior a 1 a augmentar [2].

Per això les organitzacions sanitàries mundials han publicat mesures preventives per reduir la possibilitat d'infecció. Les recomanacions són similars a les publicades per a altres coronavirus.

- Mantenir-se a distància de les altres persones, evitar viatges i activitats públiques amb gran aglomeració de persones. Controlar l'aforament en botigues i establiments.
- Rentada de mans freqüent, amb aigua i sabó o bé amb gel hidroalcohòlic.
- Tapar-se la boca en tossir o esternudar en un mocador de paper i després llençar-lo a les escombraries.
- Ús de mascaretes.

Algunes de les mesures establertes es poden controlar que es compleixin mitjançant l'ajuda de la tecnologia. Per exemple per controlar que es compleixin les limitacions d'aforament, s'utilitzi una càmera i anàlisi d'imatge, a través d'intel·ligència artificial, per detectar quantes persones entren o surten dels establiments.

En aquest projecte es buscarà la manera d'implementar una aplicació que permeti el control de l'aforament a través d'una càmera de vigilància.

1.1 Motivació

La motivació de fer aquest treball sorgeix després d'haver realitzat les practiques externes a l'empresa TissUSe GmbH de Berlín. TissUse és una empresa biotecnològica amb seu a Berlín, Alemanya, que desenvolupa "Organs-on-a-chip", com també plataformes de "Multi-Organ-Chip". Tant els "Organs-on-a-chip" com els "Multi-Organ-Chip" consisteixen una miniaturització dels òrgans per així poder estudiar l'activitat i les interaccions entre òrgans, com també les interaccions quan es presenta un estímul extern. Per així poder substituir els animals per aquests models en els assaigs clínics de prova de medicaments.

En aquella empresa juntament amb la meva companya Laura Romero, vam estar desenvolupant un software de reconeixement de cèl·lules per tal de poder-les detectar i rastrejar per comprovar la direcció a la qual es mouen dins dels microcanals del "Organ-On-a-Chip", ja que a causa d'una alta velocitat o pressió les cèl·lules podrien morir.

Amb els coneixements adquirits en aquella estada i amb una mica de recerca, hem decidit realitzar aquest TFG conjuntament, aplicant els coneixements sobre la anàlisi d'imatge a través de la llibreria Open-CV, per identificar persones i així poder controlar aforaments dels establiments mitjançant una càmera digital.

Es va decidir programar l'aplicació utilitzant dos llenguatges de programació diferents, per així poder comparar els dos programes, com també l'experiència d'utilitzar els dos llenguatges. A més, també es va optar per fer servir dos mètodes diferents per així poder veure quin era més eficient. L'aplicació desenvolupada amb *C++* faria servir un mètode de subtracció de fotogrames, i l'aplicació desenvolupada amb *Python* faria servir una xarxa neuronal per detectar persones.

2 Descripció general del projecte

A continuació es presentarà un glossari amb les paraules claus utilitzades en el treball, com també els objectius tant del projecte com de l'aplicació.

2.1 Glossari

A continuació es mostra el significat que se li ha donat a algunes paraules al llarg del document:

- Usuari: Nom genèric donat a qualsevol persona que utilitzi l'aplicació.
- Intel·ligència artificial (IA): Es refereix als sistemes o maquines que imiten la intel·ligència humana per realitzar tasques i que tenen la capacitat de millorar iterativament a partir de la informació que recopilen [3].
- Interfície gràfica (GUI): Utilització d'elements gràfics, sonors i de control per facilitar la interacció amb un sistema informàtic de manera més intuïtiva.
- Càmera de vigilància: Dispositiu o aparell que permet capturar imatges en moviment, i així realitzar un estudi a posterior o a temps real de les imatges.
- COVID-19: La malaltia causada per el virus infeccions SARS-CoV-2.
- Distancia físic de seguretat: Mesura mèdica, amb motiu per frenar la propagació d'una malaltia contagiosa, que consisteix en deixar entre 1,5 a 2 metres de distancia entre persones.
- *Confidence*: Probabilitat mínima necessària per considerar un punt com una bona detecció.
- *Capacity*: Aforament màxim que es permet dins de l'establiment.
- *Log*: Fitxer que guardarà el resultat de l'anàlisi per la seva posterior consulta.
- *Widget:* Component de la interfície gràfica que conte programació pròpia, hi ha de diferents títols per poder realitzar diferents comportaments en la GUI. Un exemple de *widget* seria un botó, o un quadre de text.
- *Csv:* És un tipus de document en format obert senzill per representar dades en forma de taula.

2.2 Objectius del projecte

L'objectiu principal d'aquest TFG és dissenyar, desenvolupar i implementar una aplicació per a ordinador que permeti controlar l'aforament d'un establiment a través de videovigilància.

Altres objectius d'aquest projecte són:

- Dissenyar i desenvolupar una interfície gràfica que permeti controlar els paràmetres de l'anàlisi.
- Permetre l'anàlisi de vídeo tant en temps real com a través d'un vídeo, ja gravat prèviament.
- Escriure els resultats de l'anàlisi en un *csv*.

- Comparar els mètodes aplicats en les dues aplicacions.

2.3 Objectius de l'aplicació

El principal objectiu de l'aplicació serà proporcionar una manera eficaç de poder controlar l'aforament.

Les funcionalitats que cobreix l'aplicació són:

- Control de l'aforament a temps real, coneixent el nombre exacte de persones a l'interior de l'establiment.
- Permetre l'anàlisi de gravacions realitzades amb anterioritat.
- Redefinir l'aforament màxim de persones en l'establiment.
- Guardar en un *csv* el tràfic total de l'establiment.

3 Requisits

A continuació es detallen les regles de negoci, els requerits funcionals i no funcionals.

3.1 Regles de negoci

01. Sempre ha d'haver-hi una càmera connectada.
02. Sempre s'ha de tenir seleccionada una opció de input.
03. Els fotogrames per segon mínim és 30 FPS.
04. El valor per defecte del paràmetre *confidence* és 0.4.
05. El valor per defecte del paràmetre *skip frames* és 30.
06. Es mostrin en temps real els usuaris a l'interior de l'establiment.
07. Es mostri els usuaris que han entrat l'establiment.
08. Es mostri els usuaris que han sortit de l'establiment.

3.2 Diagrama de casos d'us

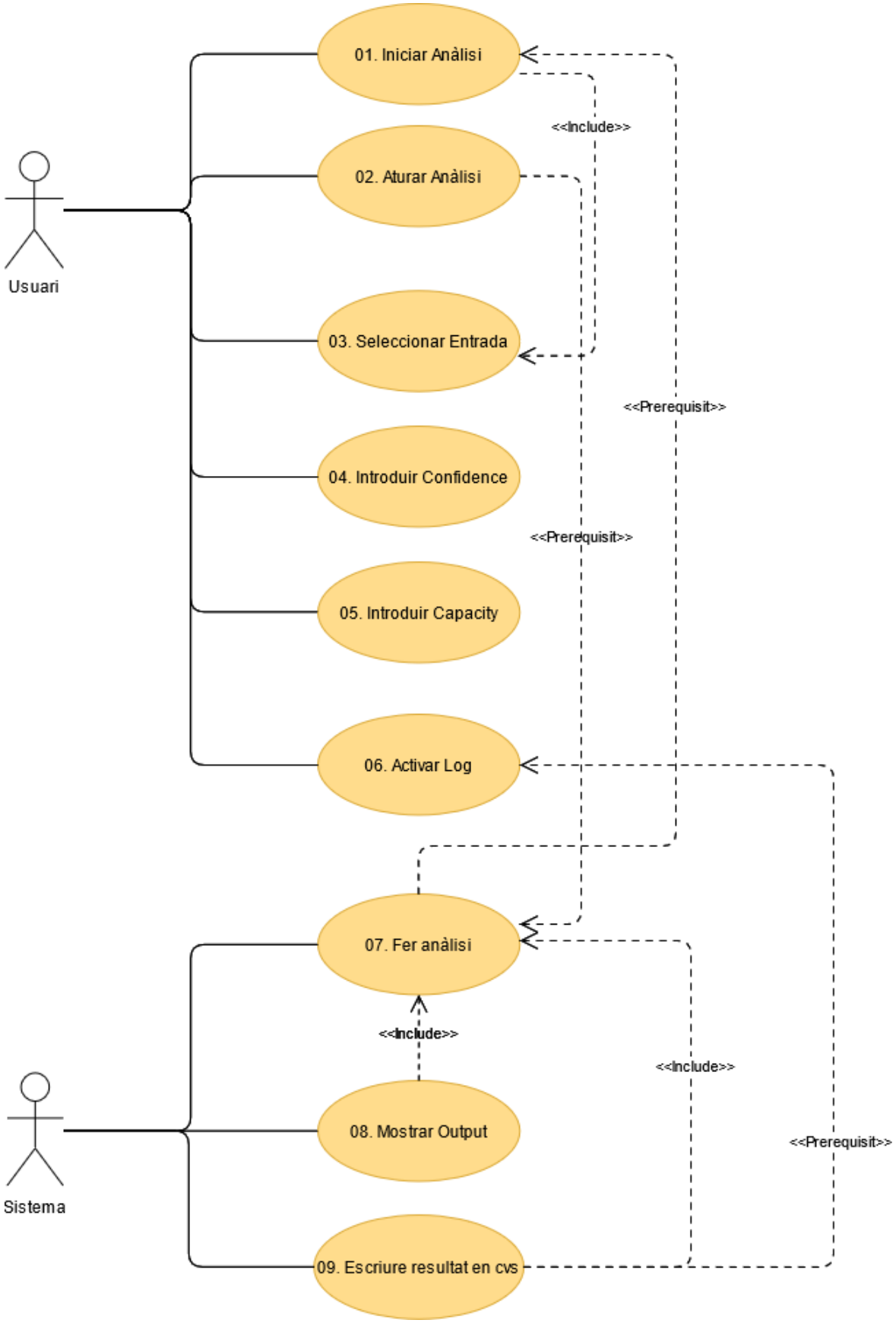

Figura 1. Diagrama de casos d'us

A la Figura 1 es mostra el diagrama de casos d'ús de l'aplicació. Aquesta figura ha estat realitzada amb el programa *Visual Paradigm* [4]. Podem observar dos actors, el propi usuari i les funcionalitats que realitza el sistema pel correcte funcionament.

3.3 Especificació dels casos d'ús

Només s'especificaran els casos d'ús de l'usuari que es mostren a la Figura 1. De manera que els casos d'ús que s'especificaran són: 01. Iniciar Anàlisi, 02. Aturar Anàlisi, 03. Seleccionar Entrada, 04. Introduir Confidence, 05. Introduir Capacity, 06. Activar Log.

3.3.1 Cdu 01. Iniciar Anàlisi

Resum de la funcionalitat: Inicialitza l'anàlisi de l'aforament de l'interior de l'establiment.

Paràmetres d'entrada: Cap.

Paràmetres de sortida: Cap.

Actors: Usuari.

Precondició: Que l'usuari hagi seleccionat i introduït una entrada.

Postcondició: Es realitzarà l'anàlisi i es mostrarà el vídeo per l'aplicació.

Procés normal principal:

1. L'usuari prem el botó "Start".
2. El sistema llegeix els valors de *Confidence* i *Capacity*. Si estan buits agafa els valors per defecte.
3. Comença l'anàlisi.
4. Es mostra el vídeo resultant per l'aplicació.

Alternatives de procés i excepcions:

1a. L'usuari prem al botó "Start" sense haver seleccionat una entrada.

1a1. El sistema espera que s'executi el cas d'ús 03. Seleccionar Entrada.

3a. L'usuari prem el botó "Stop".

3a1. El sistema atura l'anàlisi.

3b. L'usuari introdueix un valor per *Confidence*.

3b1. El sistema ignora el valor.

3c. L'usuari introdueix un valor per "*Capacity*".

3c1. El sistema ignora el valor.

3d. L'usuari activa el *Log*.

3d1. El sistema executa el cas d'ús 06. Activar Log.

3.3.2 Cdu 02. Aturar Anàlisi

Resum de la funcionalitat: Atura l'anàlisi executant-se actualment.

Paràmetres d'entrada: Cap.

Paràmetres de sortida: Cap.

Actors: Usuari.

Precondició: Que l'usuari hagi seleccionat i introduït una entrada.

Postcondició: S'atura l'anàlisi aturant també el vídeo.

Procés normal principal:

1. L'usuari prem el botó "Stop".
2. El sistema atura l'anàlisi executant-se actualment.
3. S'atura el vídeo que es mostra per l'aplicació.
4. El sistema espera que s'executi el cas d'ús 03. Seleccionar Entrada.

Alternatives de procés i excepcions:

1a. L'usuari prem al botó "Start" sense haver seleccionat una entrada.

1a1. El sistema espera que s'executi el cas d'ús 03. Seleccionar Entrada.

2a. L'usuari prem el botó "Start".

3a1. El sistema executa el cas d'ús 01. Iniciar Anàlisi.

2b. L'usuari activa el "Log".

3d1. El sistema ignora, però queda activat per la següent anàlisi.

3.3.3 Cdu 03. Seleccionar Entrada

Resum de la funcionalitat: L'usuari selecciona quin tipus d'entrada vol, si vol que s'agafi el vídeo d'una càmera o webcam, o vol que s'agafi d'un vídeo, el qual haurà de seleccionar.

Paràmetres d'entrada: *String* amb la *Ip* o valor de la càmera, o selecció del vídeo.

Paràmetres de sortida: Cap.

Actors: Usuari.

Precondició: Cap.

Postcondició: Queda guardada l'entrada, la qual es farà servir quan es pressioni el botó "Start" per fer l'anàlisi.

Procés normal principal:

1. L'usuari selecciona un dels *checkbox*, el de la càmera o el del vídeo.
    a. Si s'ha seleccionat el *checkbox* de la càmera. L'usuari introdueix una *string* amb el valor de la *Ip* de la càmera, o un "0" si vol seleccionar la webcam.

b. Si s'ha seleccionat el *checkbox* del vídeo. L'usuari selecciona en l'explorador el fitxer de vídeo.

2. Queda guardada l'entrada.
3. El sistema espera que l'usuari executi el cas d'ús 01. Iniciar anàlisi.

Alternatives de procés i excepcions:

1a. L'usuari prem al botó "Start" sense haver seleccionat una entrada.

1a1. El sistema espera que s'executi el cas d'ús 03. Selecionar Entrada.

3a. L'usuari prem el botó "Stop".

3a1. El sistema s'atura ignorant el botó.

3b. L'usuari introdueix un valor per "Confidence".

3b1. El sistema accepta el valor.

3c. L'usuari introdueix un valor per "Capacity".

3c1. El sistema accepta el valor.

3d. L'usuari activa el "Log".

3d1. El sistema executa el cas d'ús 06. Activar Log.

3.3.4 Cdu 04. Introduir Confidence

Resum de la funcionalitat: S'introdueix el valor de la variable *confidence*, com més petit sigui el valor més restrictiu és el programa, ja que es necessitaran menys *frames* per classificar un objecte com a desaparegut.

Paràmetres d'entrada: *Float* amb el valor de *confidence*.

Paràmetres de sortida: Cap.

Actors: Usuari.

Precondició: Cap.

Postcondició: Queda guardat el valor, el qual es farà servir quan es pressioni el botó "Start" per fer l'anàlisi.

Procés normal principal:

1. L'usuari pressiona l'*entry* i es mostra un cursor.
2. L'usuari introdueix el valor de *confidence*, ha de ser un *float*.
3. El valor queda guardat.
4. El sistema espera que l'usuari executi el cas d'ús 01. Iniciar anàlisi.

Alternatives de procés i excepcions:

1a. L'usuari pressiona el botó "Start", sense introduir un valor per *confidence*.

1a1. El sistema agafarà el valor per defecte de *confidence*.

2a. L'usuari intenta introduir un *string* .

2a1. El sistema no permet aquesta entrada i no deixarà escriure-la.

2b. L'usuari intenta introduir un caràcter.

2b1. El sistema no permet aquesta entrada i no deixarà escriure-la.

2c. L'usuari intenta introduir un *integer*.

2b1. El sistema no permet aquesta entrada i no deixarà escriure-la.

3.3.5 Cdu 05. Introduir Capacity

Resum de la funcionalitat: S'introdueix el valor de la variable *capacity*, com més petit sigui el valor més restrictiu és el programa, ja que es reduirà l'aforament màxim

Paràmetres d'entrada: *Integer* amb el valor de *capacity*.

Paràmetres de sortida: Cap.

Actors: Usuari.

Precondició: Cap.

Postcondició: Queda guardat el valor, el qual es farà servir quan es pressioni el botó "Start" per fer l'anàlisi.

Procés normal principal:

1. L'usuari pressiona l'*entry* i es mostra un cursor.
2. L'usuari introdueix el valor de *capacity*, ha de ser un *integer*.
3. El valor queda guardat.
4. El sistema espera que l'usuari executi el cas d'ús 01. Iniciar anàlisi.

Alternatives de procés i excepcions:

1a. L'usuari pressiona el botó "Start", sense introduir un valor per *capacity*.

1a1. El sistema agafarà el valor per defecte de *capacity*.

2a. L'usuari intenta introduir un *string* .

2a1. El sistema no permet aquesta entrada i no deixarà escriure-la.

2b. L'usuari intenta introduir un caràcter.

2b1. El sistema no permet aquesta entrada i no deixarà escriure-la.

2c. L'usuari intenta introduir un *float*.

2b1. El sistema no permet aquesta entrada i no deixarà escriure-la.

3.3.6 Cdu 06. Activar Log

Resum de la funcionalitat: Activa la funcionalitat *Log*, perquè quan es pressioni el botó "Stop", s'escrigui el resultat en un fitxer *csv*.

Paràmetres d'entrada: Cap.

Paràmetres de sortida: Cap.

Actors: Usuari.

Precondició: Cap.

Postcondició: Queda seleccionada i activada l'opció de fer el *log* al final de l'anàlisi.

Procés normal principal:

1. L'usuari prem el *checkbutton* del *log*.
2. Queda guardada que l'opció del *log* està activada.
3. El sistema espera que l'usuari executi el cas d'ús 01. Iniciar anàlisi.

Alternatives de procés i excepcions:

1a. L'usuari prem al botó "Start".

1a1. El sistema comença l'anàlisi amb l'opció del *log* activa.

1b. L'usuari prem el botó "Stop".

3a1. Si s'estava executant l'anàlisi, s'atura i es guarda el resultat.

3.4 Requisits no funcionals

Els requeriments no funcionals són els criteris que es refereixen directament a les propietats del sistema, com: eficiència, seguretat, lògica i dades, usabilitat, fiabilitat, escalabilitat.

3.4.1 Eficiència

El sistema ha de ser capaç de processar amb rapidesa cada fotograma de vídeo i actualitzar la interfície d'usuari amb el nou fotograma. Aquest refresc es produeix 30 vegades cada segon, ja que la mitjana de fotogrames per segon a la que es mostra el vídeo és de 30 FPS. Per tant el nostre programa haurà de ser ràpid amb el càlcul i la detecció de persones, per així poder refrescar adequadament la pantalla.

3.4.2 Accessibilitat

Per poder utilitzar la nostra aplicació no fa falta cap mena de registre, només una descàrrega, per tan és de fàcil accés.

Al presentar una interfície gràfica senzilla, amb explicacions de per què serveixen els diferents camps (a través de *tooltips*), permet ser utilitzada per qualsevol persona sense necessitat d'estar familiaritzat prèviament.

3.4.3 Usabilitat

El disseny de l'aplicació és intuïtiu, gràcies a la distribució adequada dels *widgets* i els *tooltips* que apareixen en els botons i en els camps de text, explicant quina funció presenta cada *widget*. Per tant és una aplicació de fàcil ús, que pot ser utilitzada per qualsevol persona que ho necessiti, sense necessitat de tindre prèviament coneixements previs per la seva utilització.

3.4.4 Fiabilitat

La tassa que es pugui produir, sigui per part d'un usuari o pel sistema, ha de ser molt baixa o inclús relativament nul·la. Per tan s'ha de limitar la capacitat d'error, per això en els camps que l'usuari ha d'omplir s'ha limitat l'entrada per nomes acceptar valors correctes.

3.4.5 Escalabilitat

Tant el disseny com la implementació de l'aplicació permeten que aquesta es pugui utilitzar en diferents sistemes operatius. Gràcies al llenguatge utilitzat per programar i l'API utilitzada per la interfície d'usuari, no hi haurà problemes de compatibilitat entre plataformes.

3.4.6 Claredat del text

L'aplicació utilitza unes mides de text i fonts adequades per la seva correcta lectura. La font utilitzada és la que trobem per defecte en les interfícies gràfiques fetes amb *Tkinter*, la font *Segoe UI*, i amb una mida fixe, ja que la mida de la finestra no es pot modificar. Frases simples i clares a l'hora d'explicar el funcionament dels *widgets*, faciliten la utilització de l'aplicació. Els colors de l'aplicació han de contrastar amb el text, per facilitar la lectura.

4 Disseny

En aquest apartat es justifica l'ús de les diferents eines i tecnologies utilitzades per dur a terme el projecte.

Per desenvolupar aquesta aplicació es volia que fos una aplicació de fàcil ús i es pogués utilitzar en diferents plataformes. Es va voler donar més importància a la interfície d'usuari, per així poder mostrar en directe la detecció i el recompte de les persones dins de l'establiment, amb una alerta en el vídeo quan se sobrepassa l'aforament màxim. També era important que la interfície fos clara, senzilla i eficaç per poder mostrar el resultat correctament.

4.1 Arquitectura de l'aplicació

4.1.1 Llenguatge

A l'hora de triar un llenguatge per poder desenvolupar l'aplicació, primerament es va estudiar quin seria el més adequat. Degut a que la nostra aplicació principal consistia en un anàlisi de vídeo per poder detectar i contar les persones, es va buscar els llenguatges més utilitzats en el vídeo anàlisi. Es va trobar que els llenguatges més utilitzats eren P*ython* [5] i *c++*. La gran utilització d'aquests dos llenguatges és a causa de que els dos són compatibles amb la llibreria d'anàlisi d'imatges *OpenCV* [6], una de les més populars i conegudes del moment.

Com és un treball compartit, es va triar repartir els llenguatges, aquest TFG, implementaria i programaria una aplicació amb P*ython*, i la meva companya Laura Romero, utilitzaria el llenguatge *c++*, per programar la seva. Així doncs es podria comparar, les aplicacions i poder observar les diferències de rendiment i eficàcia l'hora de fer l'anàlisi d'imatge, com també comparar la facilitat que presenten els dos llenguatges a l'hora de programar una aplicació amb una interfície gràfica.

Per tant el llenguatge escollit per aquest treball serà el *Python*, el *Python* es un llenguatge de programació d'alt nivell i propòsit general molt utilitzat. *Python* suporta diversos paradigmes de programació, incloent-hi la programació orientada a objectes. Presenta un sistema dinàmic i una gestió de la memòria automàtica.

Python també presenta llibreries útils que facilitaran la programació de l'aplicació, i que utilitzarem, com: *numpy* [7], *time, dlib* [8], *imutils* [9], *pil* [10], *datetime*, etc.

Figura 2. Logotip de Python

4.1.2 Entorn de desenvolupament

L'eina utilitzada com entorn pel desenvolupament ha sigut l'IDE *Visual Studio Code*. Aquest editor de codi font està desenvolupat per Microsoft per a Windows, Linux i Mac OS. Inclou suport per a la depuració, control integrat de Git, ressaltat de sintaxi, finalització intel·ligent de codi, etc. És gratuït i de codi obert. Suporta una gran quantitat de llenguatges de programació a través de complements. També trobem complements que afegeixen funcionalitats, com anàlisi de codi [11].

Figura 3. Logotip del Visual Studio Code.

4.1.3 Altres tecnologies

Per l'anàlisi i detecció de les persones s'ha utilitzat la llibreria *OpenCV*, i les xarxes neuronals *SSD detector* i *Mobilenet*.

La llibreria *OpenCV* és una llibreria lliure de visió artificial que va ser originalment desenvolupada per *Intel*. Des de la seva creació l'any 1999, s'ha utilitzat en un gran ventall d'aplicacions i fins el dia d'avui segueix sent una de les biblioteques més populars de visió artificial. Permet fer detecció de moviment, reconeixement d'objectes, reconstrucció 3D a partir d'imatge, d'entre altres funcionalitats. La seva gran popularitat és deguda a:

- Que és de software lliure, publicada sota una llicencia BSD, que permet que sigui utilitzada lliurement per propòsits tant comercials com d'investigació.
- És multiplataforma, està disponible en els sistemes operatius *GNU/Linux*, *Mac OS X*, *Windows* i *Android*, com també per diferents arquitectures de hardware.
- Està ben documentada i explicada, l'organització intenta mantenir una documentació actualitzada i tan completa com sigui possible.

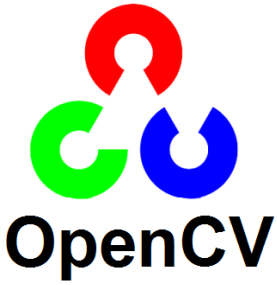

Figura 4. Logotip de la llibreria OpenCV.

MobileNet és el primer model de visió artificial de *TensorFlow*, una xarxa neuronal dissenyada per la seva utilització en mòbils o Ip càmeres, entre d'altres. Utilitza convolucions separables en profunditat per reduir el nombre de paràmetres si es compara amb una xarxa de convolucions normals. Això resulta en una xarxa neuronal profunda [12]–[14].

El *SSD detector* o *Single Shot Detector* [15] és un mètode per detectar objectes que utilitza quadres de detecció per detectar els objectes. En comptes d'utilitzar el mètode de la finestra lliscant, divideix la imatge en una taula, on en cada cel·la de la taula serà responsable de fer la detecció d'objectes en aquella regió [16].

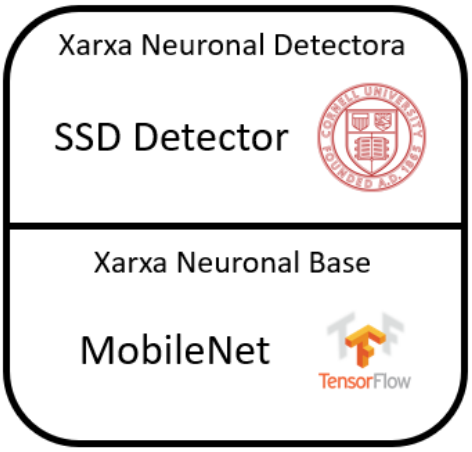

Figura 5. Representació de la combinació de les xarxes neuronals

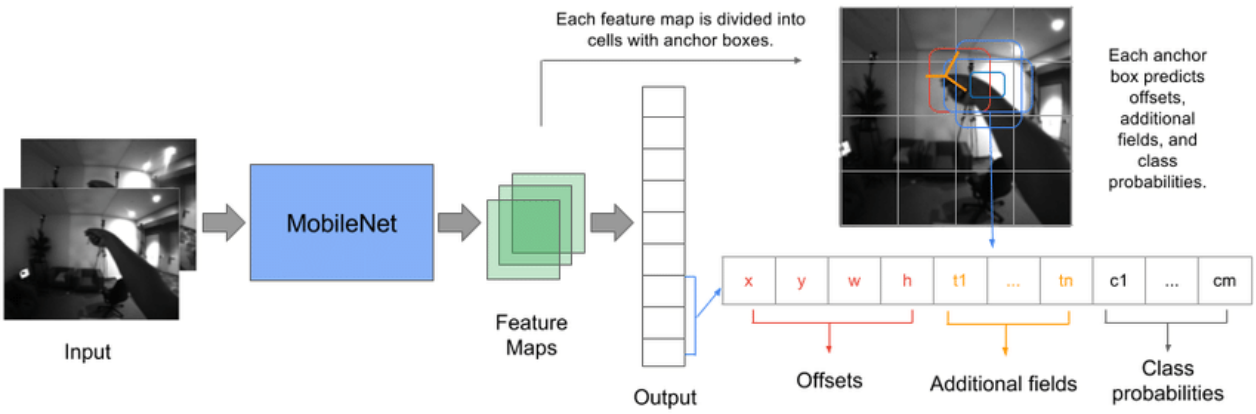

Figura 6. Utilització de *MobileNet* com a "feature extractor network" i *SSD* per la detecció [19].

La combinació d'aquestes dues xarxes neuronals permet elaborar un anàlisi exhaustiu i amb una alta probabilitat d'encert a l'hora de detectar objectes, en el nostre cas persones. S'utilitzarà la xarxa *MobileNet* per l'extracció de punts d'interès i el *SSD* com a capa de detecció dels objectes.

També s'ha utilitzat el GitHub, una eina que permet el control de versions de codi, aquesta eina conte una aplicació d'escriptori que facilita molt a l'hora de controlar les versions, per poder recuperar-les en cas de necessitat.

4.1.4 Dispositius externs

L'objectiu de la nostra aplicació és calcular en temps real l'aforament d'un establiment. Per complir el nostre objectiu necessitarem una càmera que ens transmeti el vídeo en directe de la porta de l'establiment i així poder calcular la gent que entra i surt. També és possible que es necessitin cables de connexió. A continuació explicarem tot el hardware extra necessari.

4.1.4.1 Càmeres

Les característiques recomanades per aconseguir el correcte funcionament de l'aplicació són:

- Resolució de mínim 720P.
- Una tassa mínima de 30 fotogrames per segon.

Hi ha diversos tipus de càmera que es poden utilitzar a continuació es descriuran:

- Càmera de vídeo amb connexió mitjançant Ip a través d'una xarxa wifi. Un exemple podria ser la càmera de *Wansview* de vigilància exterior. Aquesta càmera conte una lent HD de 2 Megapíxels amb una resolució fins a 1080P; connexió wifi estable de 2.4G; protecció IP66 resistent a aigua, cops i temperatures entre -10 a 40 graus Celsius; també presenta àudio bidireccional i visió nocturna.

Figura 7. Càmera *Wansview* de vigilància exterior [20].

- Càmera de fotografies connectada a una capturadora via *HDMI*. Un exemple podria ser la càmera *Nikon D3200*. Sensor *CMOS* de format DX de 24.2 Megapíxels, per obtenir imatges ben definides, amb una resolució fins a 1080P; sortida mini HDMI; micròfon ajustable; adaptador per trípode.

Figura 8. Càmera Nikon D3200 [21].

- Càmera webcam amb connexió USB. Un exemple podria ser la càmera *Logitech Brio Stream*. Presenta una resolució variable des de 720P fins a 4K; tecnologia *RightLight* 3 amb HDR, per obtenir imatges nítides; connexió mitjançant USB-C a USB-A; micròfon omnidireccional; enfocament dinàmic.

Figura 9. Càmera Logitech Brio Stream [22].

4.1.4.2 Cables i Capturadora

Algunes càmeres necessitaran estar connectades directament a través d'un cable físic l'ordinador. Per això a continuació s'especificaran els cables necessaris. A més a més algunes càmeres també necessiten l'ajuda d'una capturadora per així poder rebre correctament el vídeo en temps real.

Una de les connexions més populars és la connexió USB (figura 10), per tal que la imatge de la càmera es pugui veure en directe a la pantalla de l'ordinador o en el nostre cas a la nostra aplicació.

Figura 10. Cable USB-C a USB-A[23].

Un altre mètode per capturar la imatge d'una càmera és a través de la connexió HDMI o mini HDMI segons l'entrada del dispositiu (figura 11). Però pot ser que amb la connexió simplement no sigui suficient i per tant sigui necessari la utilització d'una capturadora d'imatge (figura 12).

Figura 11. Cable mini HDMI a HDMI [24].

Figura 12. Capturadora de vídeo HDMI [25].

4.2 Disseny de l'estructura aplicació

Respecte a l'estructura de com s'ha organitzat l'aplicació és la següent:

./directori_aplicació

- centroidtracker.py
- trackableObject.py
- peopleCounter.py
- pplCapacityCalculator.py
- tooltip.py
- deploy.caffemodel
- deploy.prototxt
- ./vídeos
  - vídeo.mp4

On cadascun dels elements duu a terme les següents funcions:

Arxiu	Funció
centroidtracker.py	La classe responsable del rastreig dels *centroids*
trackableObject.py	La classe que relaciona *centroids* amb un objecte i en guarda els valors
peopleCounter.py	La classe que s'encarrega de la detecció d'objectes
pplCapacityCalculator.py	La classe que conté la interfície, i les inicialitzacions inicials
tooltip.py	Classe que permet la incorporació dels *tooltips* en la interfície
deploy.caffemodel	Fitxer *caffe* ja pre-entrenat
deploy.prototxt	Fitxer que conte informació sobre els inputs dels fitxers d'entrada necessària per la xarxa neuronal
vídeo.mp4	Vídeo amb format mp4 per provar la detecció de persones

Taula 1. Taula per definir la funcionalitat dels fitxcrs

4.3 Disseny de la interfície

La interfície gràfica es va dissenyar amb un plantejament senzill, havia de ser una GUI que fos fàcil i intuïtiva, els *widgets* es van agrupar amb *frames*, per així tenir la GUI ordenada i fer més fàcil una futura modificació o revisió.

El disseny de la interfície es va realitzar quan ja s'havia programat l'anàlisi de persones, per així saber quins paràmetres necessitava introduir l'usuari. Aquests paràmetres eren:

- Càmera: permetre l'usuari introduir un *String* amb la *Ip* de la càmera.
- Input vídeo: permetre a l'usuari introduir un fitxer de vídeo.
- *Confidence*: permetre a l'usuari modificar el valor de *confidence*, seria la confiança de la predicció, per tant el percentatge mínim per considerar la detecció com a bona.

- *SkipFrames*: permetre a l'usuari modificar el valor, per decidir cada quant fotogrames es fa l'anàlisi.
- *Maximum capacity*: permetre modificar a l'usuari l'aforament màxim de l'establiment.

Es van dissenyar *widgets* adients amb cada paràmetre, per facilitar l'entrada d'aquests.

4.3.1 Organització de la interfície

L'organització de la GUI es va dividir amb l'ajuda de *frames* i *grids.* A continuació es mostrarà una imatge amb una representació de la interfície.

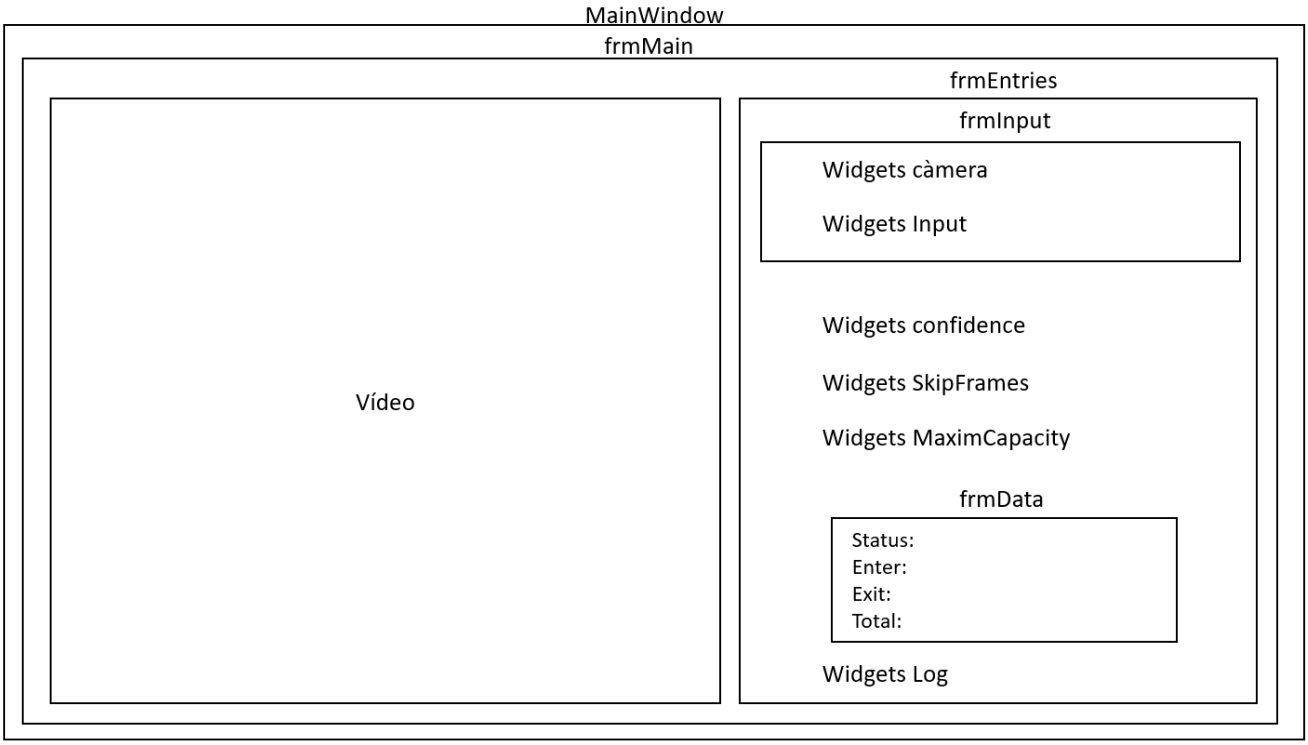

Figura 13. Representació de la interfície gràfica

Com es pot observar a la figura 1, es veu la distribució dels *widgets* i com estan estructurats i agrupats amb *frames*. Aquesta agrupació amb *frames* facilita la programació i fa més visual la distribució dels *widgets*. Primerament tenim al finestra principal amb nom *MainWindow*, a l'interior d'aquesta es té un *frame* (*frmMain*) per poder generar un *margin* entre els *widgets* i la finestra. Dins del *frame* principal trobem una label que contindrà el vídeo i un *frame* que contindrà els paràmetres que s'han de modificar per l'anàlisi.

El *frame* dels paràmetres també està dividit, tenim un *frame* per agrupar l'entrada de vídeo, sigui per càmera o per un fitxer de vídeo. A continuació trobem els *widgets* per els paràmetres per modificar la conducta de l'anàlisi. Sota dels paràmetres tenim un *frame* que conté la informació extreta de l'anàlisi, com l'estat de l'anàlisi, quantes persones han entrat i sortit de l'establiment, i el total de persones. Al final del *frmEntries* trobem un *widget* per seleccionar si es vol guardar el resultat de l'anàlisi com un *log*.

Per la implementació de la interfície d'usuari s'ha utilitzat la biblioteca *Tkinter* [17], [18], la llibreria estàndard per la implementació d'una interfície gràfica en *Python*. Es troba inclosa amb les instal·lacions de *Python* en *Linux, Windows* i *Max OS X*.

És gratuïta i compatible amb diversos sistemes operatius, presenta una manera de programar interfícies senzilla i intuïtiva, però una mica antiquada visualment, que gràcies a l'actualització i incorporació del mòdul *ttk*, s'ha adaptat a estils més moderns.

5 Implementació

Com s'ha dit anteriorment per implementar la funcionalitat de l'aplicació i la interfície d'usuari s'ha utilitzat *Pyhton* amb *Visual Studio Code.* És una aplicació amb la qual estem familiaritzats, ja que l'hem utilitzat amb anterioritat durant a la carrera.

5.1 Interfície gràfica

Per implementar la interfície gràfica he utilitzat el disseny exposat anteriorment amb la llibreria *Tkinter*. El resultat de la implementació és el següent.

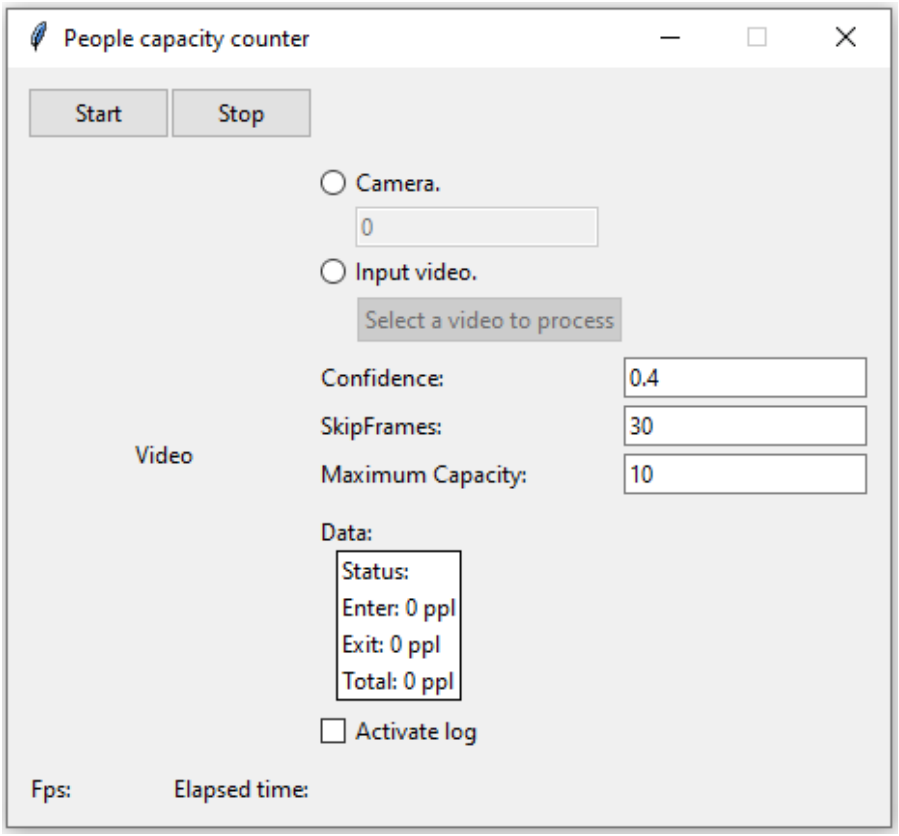

Figura 14. Finestra principal de l'aplicació

Com podem observar en la figura 14 s'ha seguit bastant l'esquema proposat, però afegint certs canvis o extres com podrien ser:

- El llenguatge de l'aplicació es l'anglès, per així poder arribar a un públic més gran, ja que es un idioma més internacional i establert com un estàndard.
- Amb cada *widget* trobem una paraula per explicar en què consisteix cada camp de text o botó.
- S'ha afegit un *tooltip* en cada *widget*, per ajudar a descriure les diferents funcionalitats del *widgets*, com es pot observar a la figura 15.

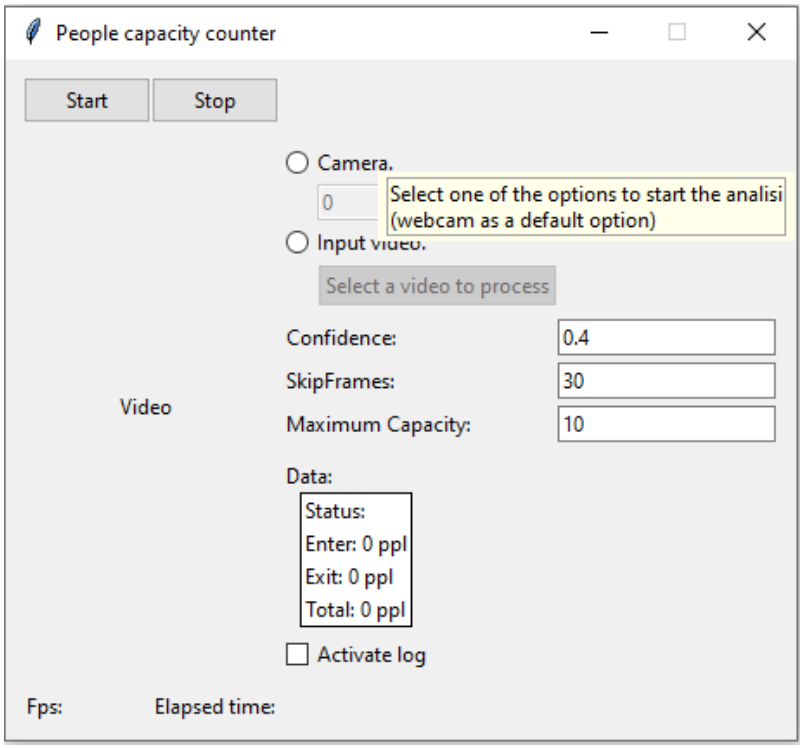

Figura 15. Finestra principal amb un tooltip.

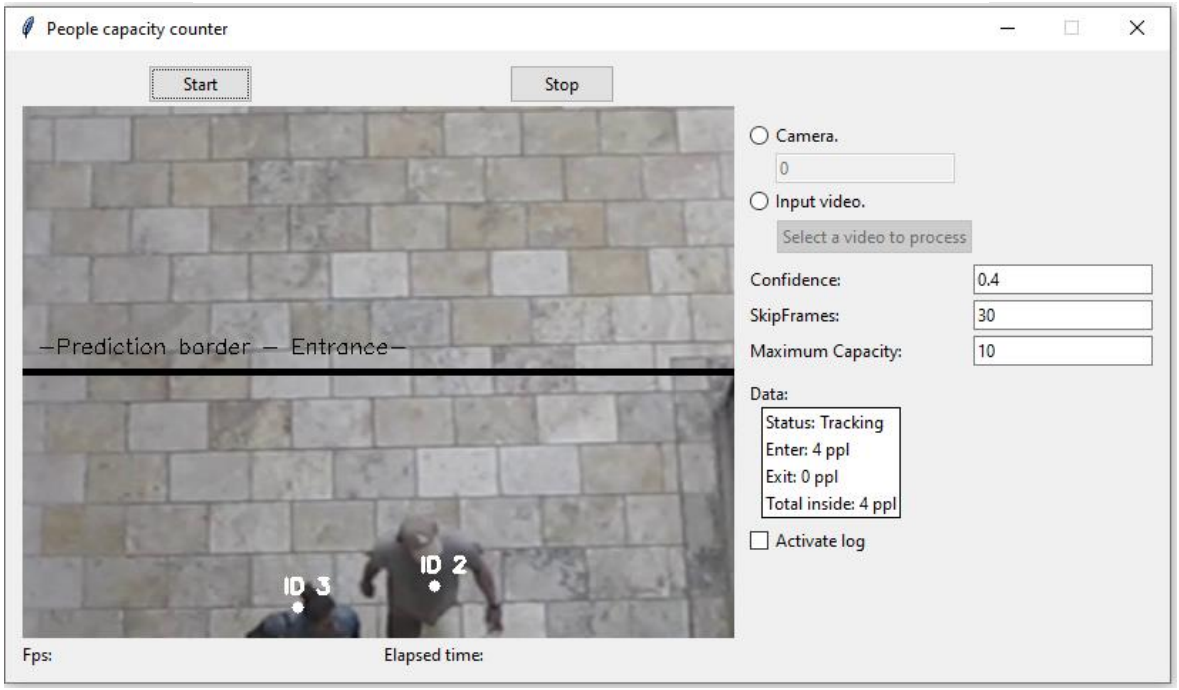

Figura 16. Finestra principal en funcionament.

Per tal de poder utilitzar el Tkinter primerament s'ha d'importar la llibreria de la següent manera:

```
...
import tkinter as tk
...
```

Un cop s'ha importat s'ha de crear la finestra en el procés main i modificar els paràmetres desitjats, en el nostre cas modificar el nom i fer que no es pugui modificar la mida. Després definir la nostra pròpia finestra i mantenir-la oberta fins que es tanqui.

```
 ...
 root = tk.Tk()
 root.title("People capacity counter")
 root.resizable(False, False)
 app = pplCApp(root)
 root.mainloop()
...
```

Primer agafarem la finestra principal, i després per afegir elements fa falta definir quin serà el frame principal i a continuació afegir els elements en aquest frame, com per exemple:

```
...
 self.master = master
 # Main Frame, enables a margin
 self.frmMain = tk.Frame(self.master, padx=10, pady=10)
...
```

Com podem observar al crear el frame el primer camp, s'ha d'introduir en quin widget contenidor es trobarà, i com a camps extres es poden modificar les propietats que aquest frame tindrà.

A continuació es posaran exemples de definició per cada widget, on en el primer camp es trobarà el widget contenidor i com a camps extres les propietats a modificar, com en el frame anterior. Les definicions són dels següents widgets: Button, Label, Entry

Radiobutton.

```
    ...
    self.btStart = ttk.Button(self.frmMain, text="Start",
command=self.inicount)
    self.lblVideo = ttk.Label(self.frmMain, text="Video",
padding=(0,0,5,0))
    self.entCamera = ttk.Entry(self.frmInput,
textvariable=self.varCamera)
    self.radiobtCamera = ttk.Radiobutton(self.frmInput, text="Camera.",
value=0, command=lambda e1=self.entCamera, e2=self.btFileDialog:
self.valcheck(e1, e2))
    ...
```

Per obtenir el valor que l'usuari hauria introduït al widget, s'han definit variables del tk que obtindrien el valor introduït en el camp. Les definicions d'aquestes són:

```
    ...
    self.varConfidence = tk.StringVar()
    #self.varConfidence = "0.4"
    self.lblConfidence = ttk.Label(self.frmEntries, text="Confidence:")
    self.entConfidence = ttk.Entry(self.frmEntries, validate='key',
validatecommand=(self.frmEntries.register(self.validateEntryFloat),
'%P'), textvariable=self.varConfidence)
    ...
```

Com podem veure per assignar una variable al widget s'ha d'utilitzar la propietat textvariable, a més a més si es vol fer una validació de què el valor sigui correcta s'hi ha d'assignar una funció a la propietat validatecommand perquè determini si el valor es correcte o no. Un exemple d'aquesta funció seria:

```
    ...
    # Function to only allow float in the entry
    def validateEntryFloat(self, ent):
            try:
                float(ent)
            except:
                return False
            return True
    ...
```

Un cop definits cada *widget* amb les propietats i comportaments desitjats, s'han de compilar amb la funció *pack()* o *grid()*. En el nostre cas utilitzarem majoritàriament la funció *grid()* ja que et permet distribuir els *widgets* amb un format de taula. Aquestes dues funcions també permeten modificar les propietats i comportaments dels *widgets* relacionats

amb el posicionament, com afegir marge, que siguin extensibles. A més a més com s'ha dit anteriorment la funció *grip()* permet distribuir els *widgets* en format taula i així definir en quina columna o fila es troben com si ocupen més d'una línia o columna. Un exemple de cada definició seria:

```
...
self.frmMain.pack(fill=tk.BOTH, expand=True)
self.lblVideo.grid(column=0, row=1, columnspan=2, rowspan=6)
...
```

Per afegir la funcionalitat del *tooltip* s'ha hagut d'agregar una classe al projecte, creada per la comunitat de programadors *Tkinter*. Un cop afegida la classe s'ha de crear una entitat d'aquesta i relacionar-la amb el *widget* desitjat, en el primer camp i com a segon el text que es vol que es mostri.

```
...
Tooltip(self.radiobtCamera, text='Select one of the options to start
the analisi (webcam as a default option)')
...
```

5.2 Funcionalitat aplicació

A continuació es detallarà com s'han implementat les diferents funcionalitats de l'aplicació fitxer per fitxer.

5.2.1 Implementació del fitxer pplCapacity.py

Primerament començarem pel fitxer ***pplCapacityCalculator.py***, fitxer que conté la implementació de la interfície gràfica i les inicialitzacions necessàries per poder dur a terme l'anàlisi.

Primerament s'agafen els paràmetres introduïts per la interfície, com estan definits com a variables especials per ser tractades amb *Tkinter*, s'utilitzarà el mètode *get()* per treure'n el valor, un exemple seria:

```
...
maximum = self.varMaximum.get()
confidenceArg = float(self.varConfidence.get())
...
```

Si en canvi se'n vol treure el valor del text del *widget* directament el mètode canviaria, i s'hauria de fer de la següent manera:

```
...
inputArg = self.btFileDialog['text']
...
```

A continuació és necessari inicialitzar la xarxa neuronal, carregant el model ja pre-entrenat com el fitxer d'informació addicional necessari. A més a més es crearà un *VideoStream* per capturar l'input de la càmera, o un *VideoCapture* si es rep com a entrada un fitxer de vídeo.

```
...
# Load our model
        net = cv2.dnn.readNetFromCaffe(prototxtArg, modelArg)
        # If a video path was not supplied, grab the ip camera or
the webcam
        if inputArg == "Select a video to process":
            print("[INFO] Starting the live stream.. in "+cameraArg)
            # http://192.168.1.45:8080/video
            if "http" in cameraArg:
                self.vs = VideoStream(cameraArg).start()
            else:
                self.vs = VideoStream(int(cameraArg)).start()
        # Otherwise, grab a reference to the video file
        else:
            print("[INFO] Starting the video..")
            self.vs = cv2.VideoCapture(inputArg)
...
```

Agafem el frame del vídeo per analitzar-lo:

```
...
# Loop over frames from the video stream
        while True :
            # Grab the next frame and handle if we are reading from
either VideoCapture or VideoStream
            frame = self.vs.read()
            frame = frame[1] if (inputArg != "Select a video to
process") else frame
...
```

Cridem a la funció per que ens calculi els objectes.

```
...
    # Call the main function to calculate the objects that had moved
donw or up
        frame, totalUp, totalDown, empty, empty1, total, trackers,
status   = pplC.countPPl(frame, W, H, totalFrames, skipFramesArg, net,
confidenceArg, CLASSES, ct, trackableObjects, totalUp, empty, totalDown,
empty1, trackers, total, maximum)
    ...
```

Un cop es retornen els càlculs de l'anàlisi és controlar que no es superi l'aforament, en cas que se superi aquest s'escriu una alerta en el vídeo.

```
    ...
    # If the people limit exceeds over threshold
    if total >= maximum:
        cv2.putText(frame, "ALERT: Limit of people exceeded", (10,
frame.shape[0] - 80), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 255), 2)
    ...
```

I s'imprimeixi per pantalla el *frame*, i actualitzem la interfície gràfica perquè mostri els valors resultants de l'anàlisi.

```
    ...
    # Print the frame in the window
                if np.shape(frame) != ():
                    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                    im =Image.fromarray(rgb)
                    img = ImageTk.PhotoImage(image=im)
                    self.lblVideo.configure(image=img)
                    self.lblVideo.after(1)
                    self.lblVideo.update()

                # Show the information of the calculation
                self.lblStatus.configure(text="Status: "+status)
                self.lblEnter.configure(text="Enter:{}ppl"
    .format(totalDown))
                self.lblExit.configure(text="Exit:{}ppl"
    .format(totalUp))
                self.lblTotal.configure(text="Total inside: {} ppl"
    .format(total))
```

5.2.2 Implementació del fitxer peopleCounter.py

A continuació explicarem les parts del codi més importants del fitxer ***peopleCounter.py***. Aquest fitxer bàsicament conté la classe *pplCounter* que s'encarrega d'identificar i contar les persones que apareixen al *frame* del vídeo.

Es comença realitzant les preparacions necessàries, com reajustar la mida del frame o modificar-li el format de color. I es Comença amb l'anàlisi, primerament es transforma en un Blob, un format de per emmagatzemar informació, necessari per utilitzar la xarxa neuronal. I es comença amb la detecció d'objectes.

```
...
# Convert the frame to a blob and pass the blob through the
# network and obtain the detections
blob = cv2.dnn.blobFromImage(frame, 0.007843, (W, H), 127.5)
net.setInput(blob)
detections = net.forward()
...
```

Filtrem les deteccions per sobre del mínim de *confidence* i mirem si el que s'ha detectat ha sigut una persona.

```
...
# Extract the confidence associated with the prediction
 confidence = detections[0, 0, i, 2]
 #Filter out weak detections by requiring a minimum confidence
 if confidence > confidenceArg:
       # Extract the index and the label, from the detections list
       idx = int(detections[0, 0, i, 1])
       # If the class label is not a person, ignore it
       if CLASSES[idx] != "person":
                  continue
...
```

A continuació es dibuixa un rectangle al voltant de la persona detectada i es guarda per poder fer-li el seguiment posteriorment.

A continuació si no es detecten més persones es passa a fer el seguiment. Es dibuixa una lineal al mig per poder dir si entren o surten del local, ja que la línia dibuixada representa la porta.

```
...
# Draw a horizontal line in the center of the frame -- once an
# object crosses this line we will determine whether they were
# moving 'up' or 'down'
cv2.line(frame, (0, H // 2), (W, H // 2), (0, 0, 0), 3)
cv2.putText(frame, "-Prediction border - Entrance-", (10, H - ((i *
20) + 200)), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1)
...
```

Es fa un bucle per iterar per les persones detectades i així actualitzar la seva posició i en cas de travessar la línia, contar aquella persona com si hagués entrat o sortit, segons el sentit en què creua la línia.

```
...
# Loop over the tracked objects
for (objectID, centroid) in objects.items():
	# The difference between the y-coordinate of the *current*
	# centroid and the mean of *previous* centroids will tell
	# us in which direction the object is moving (negative for
	# 'up' and positive for 'down')
	y = [c[1] for c in to.centroids]
	direction = centroid[1] - np.mean(y)
	to.centroids.append(centroid)
...
```

Contar l'objecte si està entrant o està sortint de l'establiment i es calcula el total a l'interior de l'establiment.

```
...
# If the direction is negative (indicating the object
 # is moving up) AND the centroid is above the center
 # line, count the object
 if direction < 0 and centroid[1] < H // 2:
	totalUp += 1
	empty.append(totalUp)
	to.counted = True
```

```
 # If the direction is positive (indicating the object
 # is moving down) AND the centroid is below the
 # center line, count the object
 elif direction > 0 and centroid[1] > H // 2:
       totalDown += 1
       empty1.append(totalDown)

# Compute the sum of total people inside
                        total = len(empty1)-len(empty)
...
```

Es retornen els valors calculats de l'anàlisi.

```
...
return frame, totalUp, totalDown, empty, empty1, total, trackers,
status
```

5.2.3 Implementació del fitxer centroidtracker.py

En aquest fitxer trobem la classe *CentroidTracker*, la funció d'aquesta classe és principal d'emmagatzemar l'objecte i relacionar el requadre que l'envolta amb el *centroid*.

Compte una funció per registrar el centroide i així poder guardar-lo en un diccionari, com també una funció una per desregistrar, eliminant la relació i l'entrada del diccionari.

```
...
def register(self, centroid):
       # when registering an object we use the next available object
       # ID to store the centroid
       self.objects[self.nextObjectID] = centroid
       self.disappeared[self.nextObjectID] = 0
       self.nextObjectID += 1
def deregister(self, objectID):
       # to deregister an object ID we delete the object ID from
       # both of our respective dictionaries
       del self.objects[objectID]
       del self.disappeared[objectID]
...
```

Com també necessitem mantenir actualitzada en tot moment la localització de cada centroide, s'ha implementat un mètode per fer-ho. Primerament mirem si tenim punts, si no en tenim, doncs els marquem com a desapareguts i si superen el màxim nombre de *frames* desapareguts consecutivament els desregistrem.

```
...
if len(rects) == 0:
        # loop over any existing tracked objects and mark them
        # as disappeared
        for objectID in list(self.disappeared.keys()):
              self.disappeared[objectID] += 1
              # if we have reached a maximum number of consecutive
              # frames where a given object has been marked as
              # missing, deregister it
              if self.disappeared[objectID] > self.maxDisappeared:
                    self.deregister(objectID)

              #return early as there are no centroids or tracking
info to update
              return self.objects
...
```

Si tenim objectes hem de calcular el *centroid* a partir del rectangle que envolta l'objecte. I registrar-los si es considera un objecte nou o si ja existeix, doncs actualitzar la posició actual i relacionar-lo amb un *centroid* del *frame* anterior. En el cas de tindre un nombre diferent de *centroids* entre el *frame* anterior i actual, repassem els *centroids* que no s'han pogut relacionar i en cas d'haver superat el màxim de *frames* que pot estar desaparegut, el esborrem del registre.

5.2.4 Implementació del fitxer trackableObject.py

El fitxer *trackableObject.py* conte la classe *TrackableObject*, que emmagatzema la ID del objecte amb el seu *centroid*. I una variable control.

```
...
class TrackableObject:
 def __init__(self, objectID, centroid):
       # store the object ID, then initialize a list of centroids
       # using the current centroid
       self.objectID = objectID
       self.centroids = [centroid]
       # initialize a boolean used to indicate if the object has
       # already been counted or not
       self.counted = False
...
```

6 Avaluació

Per avaluar el correcte funcionament de l'aplicació es va realitzar un Joc de proves, que avaluava les diferents funcionalitats, com també la resposta de la interfície gràfica.

Prova	Resultat esperat	Resultat
Obrir l'aplicació.	S'obre l'aplicació i es mostra la interfície gràfica.	Correcte
Es fa clic sobre el botó de "Stop" sense haver primerament fet clic al botó "Start" .	No passa res, es segueix esperant que es faci clic al botó "Start".	Correcte
Es fa clic al botó "Start" sense haver afegit cap paràmetre d'entrada.	S'executa el programa agafant per entrada la càmera per defecte del ordinador on s'executa.	Correcte
Al seleccionar un vídeo com entrada s'intenta seleccionar un fitxer que no es de format .mp4.	No es dona aquesta opció, ja que nomes es poden veure i seleccionar els fitxers amb format .mp4.	Correcte
En el paràmetre *confidence* s'intenta introduir un caràcter.	No es pot, ja que quan es va a introduir un caràcter no s'escriu en el camp.	Correcte
En el paràmetre *skipFrames* s'intenta introduir un caràcter.	No es pot, ja que quan es va a introduir un caràcter no s'escriu en el camp.	Correcte
En el paràmetre *Maximum capacity* s'intenta introduir un caràcter.	No es pot, ja que quan es va a introduir un caràcter no s'escriu en el camp.	Correcte
En el paràmetre *skipFrames* s'intenta introduir un paràmetre decimal.	No es pot, ja que quan es va a introduir el punt (considerat un caràcter) no s'escriu en el camp.	Correcte
En el paràmetre *Maximum capacity* s'intenta introduir un paràmetre decimal.	No es pot, ja que quan es va a introduir el punt (considerat un caràcter) no s'escriu en el camp.	Correcte
S'activa el *log*.	Es guardarà en un fitxer el recompte total de gent.	Correcte
Entrada: Vídeo. Es detecten correctament les persones (Figura 17).	S'assigna una ID i s'identifica en la pantalla a la persona.	Correcte
Entrada: Vídeo. Es detecten correctament si varies persones van molt juntes (Figura 18).	S'assigna una ID a cada persona i s'identifica cada persona en la pantalla.	Correcte
Entrada: Vídeo. Es detecten correctament la sortida de persones (Figura 19).	S'assigna una ID i s'identifica en la pantalla a la persona, i es calcula el sentit del moviment.	Correcte

Entrada: Vídeo. Es detecten correctament l'entrada de persones (Figura 17).	S'assigna una ID i s'identifica en la pantalla a la persona, i es calcula el sentit del moviment.	Correcte
Entrada: Vídeo. S'actualitza el comptador d'entrada amb l'entrada de persones.	S'incrementa en un el comptador d'entrada.	Correcte
Entrada: Vídeo. S'actualitza el comptador de sortida amb la sortida de persones.	S'incrementa en un el comptador de sortida.	Correcte
Entrada: Vídeo. S'actualitza el aforament total amb l'entrada i sortida de persones.	S'actualitza correctament el aforament actual, restant una persona si es surt, sumant una persona si s'entra	Correcte
Entrada: Càmera. Es detecten correctament les persones (Figura 20).	S'assigna una ID i s'identifica en la pantalla a la persona.	Correcte
Entrada: Càmera. Es detecten correctament si dos persones van molt juntes (Figura 22).	S'assigna una ID a cada persona i s'identifica cada persona en la pantalla.	Correcte
Entrada: Càmera. Es detecten correctament la sortida de persones (Figura 21).	S'assigna una ID i s'identifica en la pantalla a la persona, i es calcula el sentit del moviment.	Correcte
Entrada: Càmera. Es detecten correctament l'entrada de persones (Figura 20).	S'assigna una ID i s'identifica en la pantalla a la persona, i es calcula el sentit del moviment.	Correcte
Entrada: Càmera. S'actualitza el comptador d'entrada amb l'entrada de persones.	S'incrementa en un el comptador d'entrada.	Correcte
Entrada: Càmera. S'actualitza el comptador de sortida amb la sortida de persones.	S'incrementa en un el comptador de sortida.	Correcte
Entrada: Càmera. S'actualitza el aforament total amb l'entrada i sortida de persones.	S'actualitza correctament el aforament actual, restant una persona si es surt, sumant una persona si s'entra	Correcte
Entrada: Càmera. Detecció correcte quan una persona entra i una altre surt (Figura 23).	S'assigna una ID a cada persona i s'identifica a la pantalla. S'actualitza el aforament actual	Correcte

Taula 2. Taula de Joc de Proves**.**

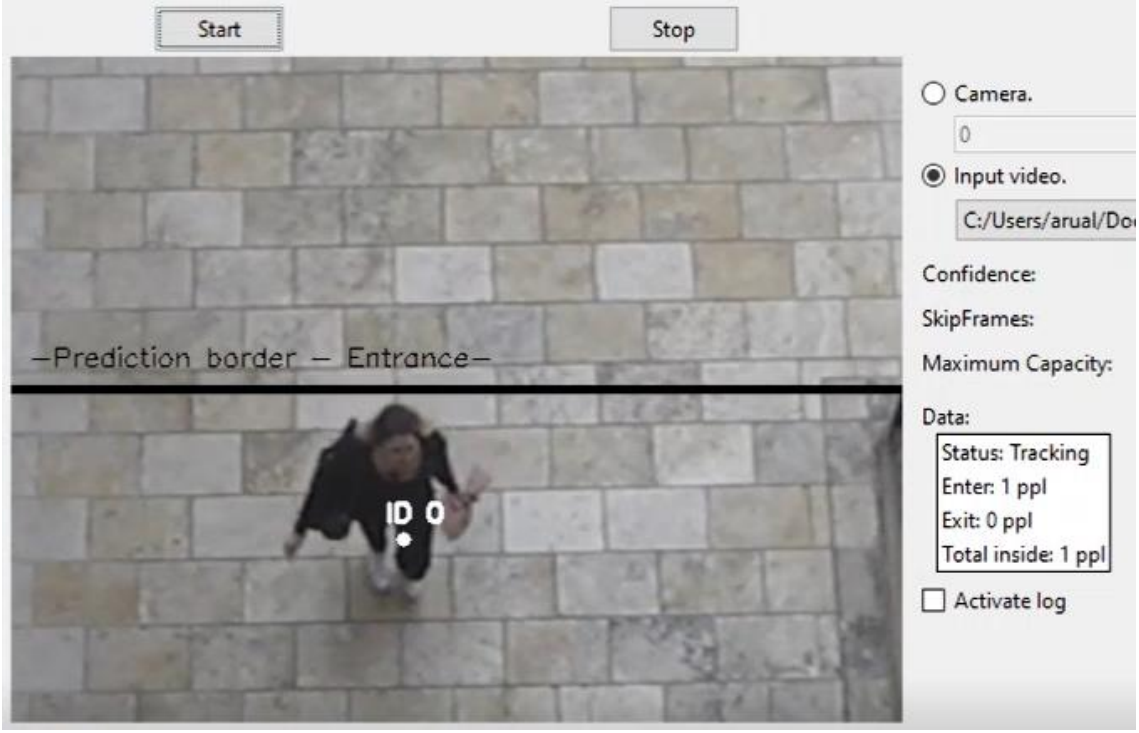

Figura 17. Detecció d'una persona entrant.

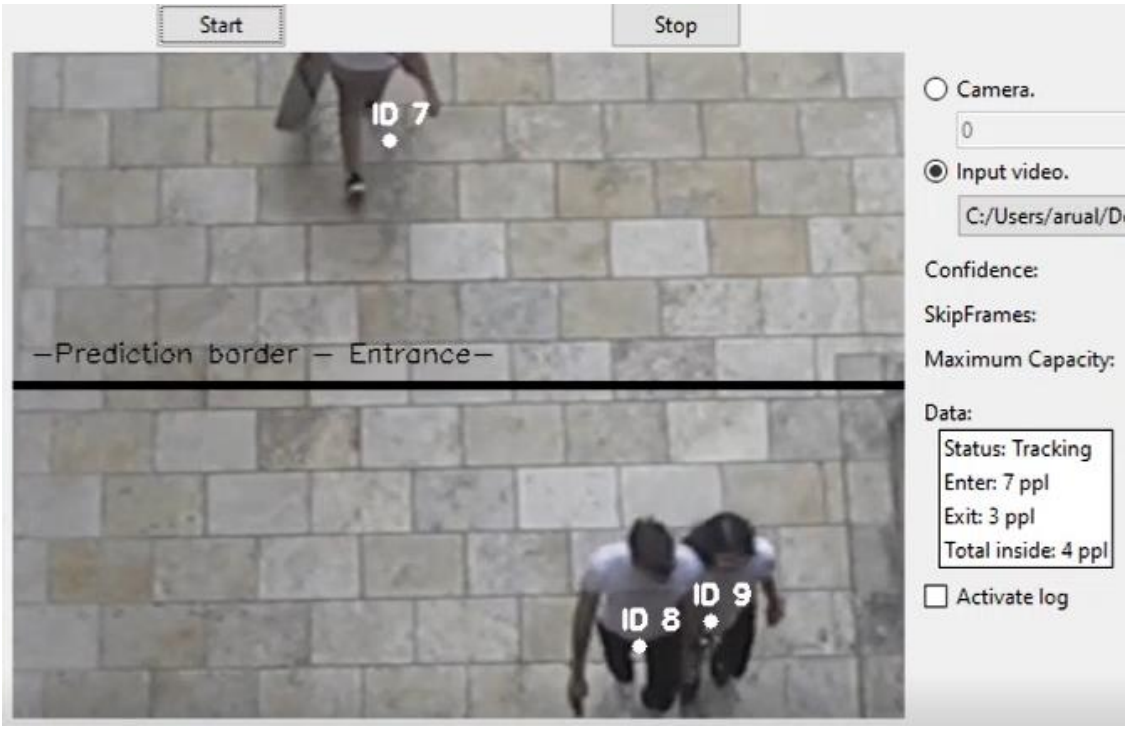

Figura 18. Detecció de varies persones.

Figura 20. Detecció de varies persones sortint.

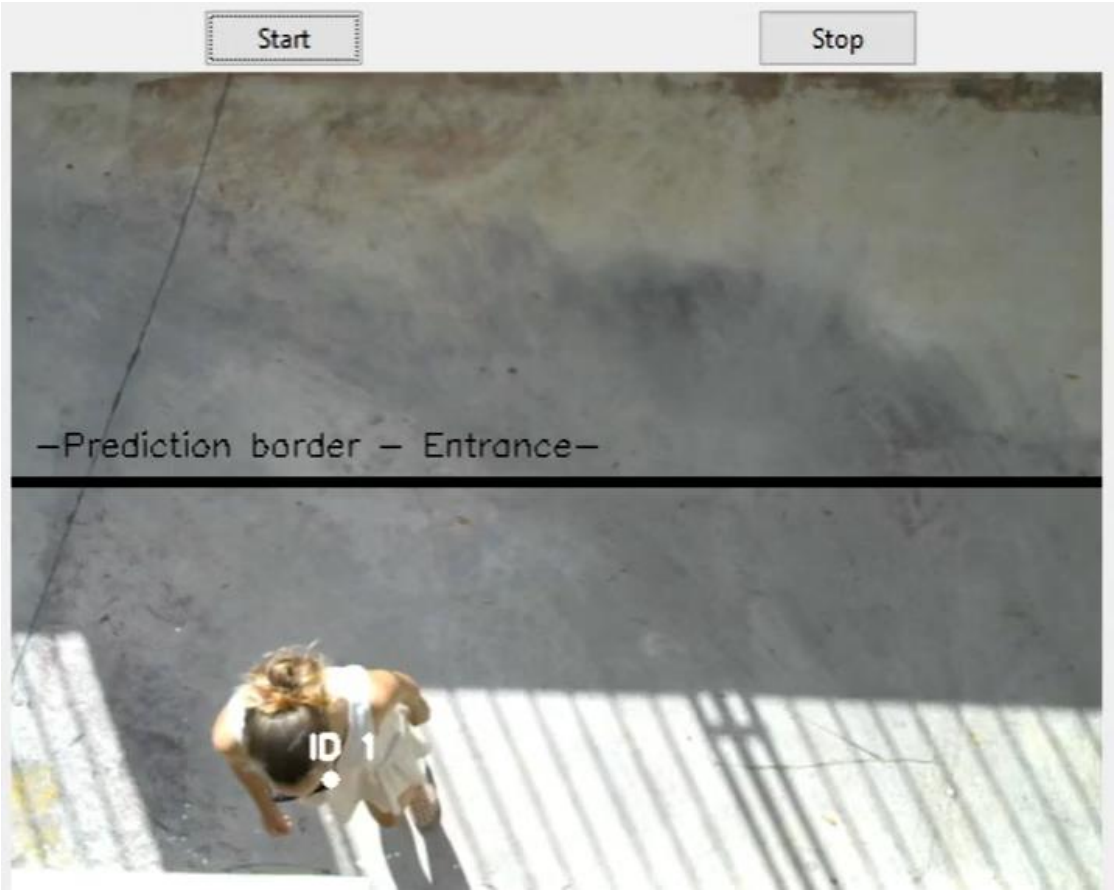

Figura 19. Detecció d'una persona entrant.

Figura 21. Detecció d'una persona sortint.

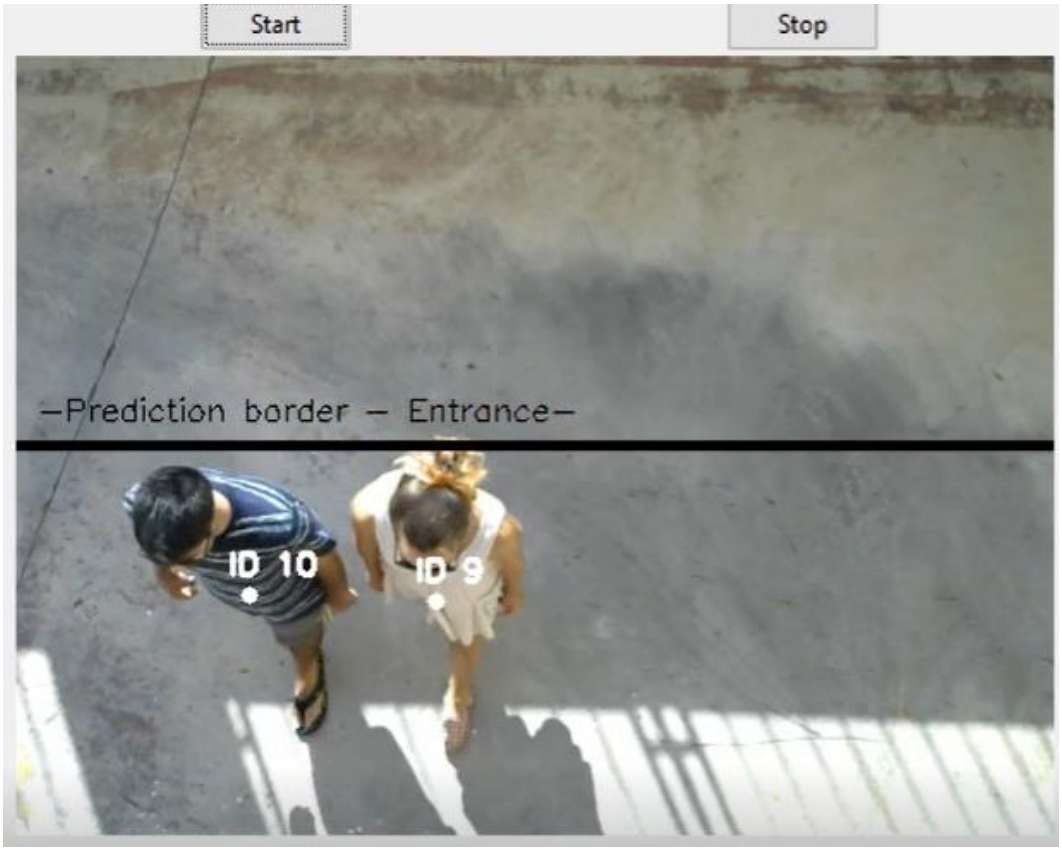

Figura 22. Detecció de varies persones juntes.

Figura 23. Detecció de una persona entrant i una altre sortint.

Figura 24. Detecció de una persona entrant en l'aplicació de C++.

7 Conclusions

7.1 Conclusions del funcionament de l'aplicació

Com a conclusions després d'haver programat i avaluat l'aplicació podem dir que compleix totes les funcionalitats esperades. Es detecten correctament les persones amb l'anàlisi i per tant es pot calcular si les persones entraven o sortien de l'establiment, entenent com la línia al mig de la pantalla l'entrada de l'establiment. També es pot modificar el comportament de l'aplicació a través de la modificació dels paràmetres de la interfície gràfica.

L'aplicació ha estat provada amb un vídeo de prova, però per poder provar la funcionalitat en temps real amb una càmera no ha sigut fàcil trobar una bona localització. Finalment s'ha realitzat el joc de proves amb una càmera localitzada a una alçada de 4m respecte al terra. Ja que pel correcte funcionament de l'aplicació es necessitava una vista aèria.

Al fer el treball conjuntament amb la meva companya Laura Romero hem pogut comparar els resultats obtinguts qualitativament. S'ha pogut observar que el programa de *Python* era més acurat a l'hora de la detecció de persones, gracies a l'anàlisi mitjançant xarxes neuronals, en canvi l'aplicació realitzada amb *C++* era més sensible i a vegades detectava objectes sense ser persones com per exemple en la Figura 24, i les ombres podien dificultar la detecció. Tot i això com els objectes que es detectaven eren estàtics, no interferien en el control de l'aforament.

Si fem una comparació de l'experiència d'ús de l'aplicació per part de l'usuari, s'han obtingut resultats força similars. Es va preguntar a diferents amics i familiars que provessin l'aplicació i que ens donessin la seva opinió, per saber quina resultava més senzilla o tenia problemes en algun aspecte.

Com a primer aspecte tenim la selecció i edició dels paràmetres desitjats. En cap de les dues aplicacions és necessari, ja que poden funcionar en paràmetres per defecte. En el cas que no es vulgui modificar els paràmetres, l'aplicació en *C++* seria la més desitjada, ja que no permet la modificació d'aquests i per tant la utilització pot ser més senzilla. En canvi l'aplicació en *Python* permet la modificació d'aquests i pot ser utilitzada per persones amb certa experiència i que vulguin optimitzar més el funcionament de l'aplicació.

El segon aspecte on ens centrarem és la dificultat per iniciar la visualització de la càmera en directe, en les dues aplicacions es força senzill. On la principal diferencia, és que després de fer clic al botó d'inici, en l'aplicació de *C++* es tanca la finestra actual i apareix una nova finestra amb la visualització de la càmera, en canvi en l'aplicació realitzada en *Python* es mostra en la mateixa finestra. En cap cas es va tenir dificultat per iniciar la càmera.

Un cop el programa està en funcionament és necessari mostrar la informació sobre l'aforament en pantalla. En l'aplicació en *Python* tenim tota la informació en un requadre a la dreta de la imatge. Per contra en l'aplicació en *C++* es mostra la informació en el mateix vídeo. A més a més en l'aplicació en *C++* es mostra la quantitat de persones que poden entrar en l'aplicació, al contrari que l'aplicació en *Python* que et mostra les persones que han entrat i sortit. Es va coincidir que la informació mostrada per l'aplicació en *C++* era més intuïtiva en mostrar les persones que poden entrar i no requeria fer el càlcul per saber quanta gent hi ha dins de l'establiment. L'aplicació en *Python* presentava un sistema d'alerta quan l'aforament s'havia sobrepassat, cosa que l'aplicació en *C++* no ho

presentava i permetia informar quan s'havia de prendre alguna mesura per tornar a estar dins de l'aforament permès.

Finalment es va mesurar el tancament de l'aplicació, en l'aplicació en *Python* podies parar el vídeo si simplement es volia parar l'anàlisi, i per tancar l'aplicació es podia prémer la creu vermella de la finestra. En canvi per tancar l'aplicació en *C++* s'havia de prémer la tecla "esc", cosa que podia portar algun entrebanc.

7.2 Conclusions experiència personal

El desenvolupament d'aquesta aplicació m'ha permès aprendre amb major profunditat a programar una aplicació d'anàlisi d'imatges i introduir-me en el món del càlcul mitjançant intel·ligència artificial. Personalment m'ha semblat un projecte molt interessant que m'ha permès aplicar els coneixements adquirits durant les pràctiques externes realitzades a *TissUse GmbH,* Berlín, tot i que per una àrea tècnica diferent.

Tot i les dificultats que s'han anat trobant durant el desenvolupament del programa, crec que l'aplicació ha presentat un bon disseny final, amb una interfície gràfica simple i entenedora, per així facilitar el seu ús i que no es requereixi persones amb un alt coneixement per ser utilitzada.

En general m'ha semblat un projecte molt enriquidor i una bona experiència dins de l'àmbit de desenvolupament de software, com en el d'anàlisi digital.

8 Referències

[1] W. K. Baek, S.-Y. Sohn, A. Mahgoub, i R. Hage, «A Comprehensive Review of Severe Acute Respiratory Syndrome Coronavirus 2», *Cureus*, maig 2020, doi: 10.7759/CUREUS.7943.

[2] «COVID-19 - Viquipèdia, l'enciclopèdia lliure». https://ca.wikipedia.org/wiki/COVID-19#Prevenció (accedit ago. 09, 2021).

[3] «What is Artificial Intelligence (AI)? | IBM». https://www.ibm.com/cloud/learn/what-is-artificial-intelligence (accedit ago. 09, 2021).

[4] «Ideal Modeling & Diagramming Tool for Agile Team Collaboration». https://www.visual-paradigm.com/ (accedit ago. 10, 2021).

[5] «Welcome to Python.org». https://www.python.org/ (accedit ago. 15, 2021).

[6] «Home - OpenCV». https://opencv.org/ (accedit ago. 15, 2021).

[7] «NumPy». https://numpy.org/ (accedit ago. 15, 2021).

[8] «dlib · PyPI». https://pypi.org/project/dlib/ (accedit ago. 15, 2021).

[9] «imutils · PyPI». https://pypi.org/project/imutils/ (accedit ago. 15, 2021).

[10] «Pillow — Pillow (PIL Fork) 8.3.1 documentation». https://pillow.readthedocs.io/en/stable/ (accedit ago. 15, 2021).

[11] «Visual Studio Code - Code Editing. Redefined». https://code.visualstudio.com/ (accedit ago. 15, 2021).

[12] «MobileNet Convolutional neural network Machine Learning Algorithms | Analytics Vidhya». https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470 (accedit ago. 14, 2021).

[13] X. Feng, R. Xie, J. Sheng, i S. Zhang, «Population Statistics Algorithm Based on MobileNet», *J. Phys. Conf. Ser.*, vol. 1237, núm. 2, p. 022045, juny 2019, doi: 10.1088/1742-6596/1237/2/022045.

[14] «GitHub - chuanqi305/MobileNet-SSD: Caffe implementation of Google MobileNet SSD detection network, with pretrained weights on VOC0712 and mAP=0.727.» https://github.com/chuanqi305/MobileNet-SSD (accedit ago. 14, 2021).

[15] W. Liu *et al.*, «SSD: Single shot multibox detector», *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9905 LNCS, p. 21-37, 2016.

[16] «How single-shot detector (SSD) works? | ArcGIS Developer». https://developers.arcgis.com/python/guide/how-ssd-works/ (accedit ago. 15, 2021).

[17] «tkinter — Python interface to Tcl/Tk — Python 3.9.6 documentation». https://docs.python.org/3/library/tkinter.html (accedit ago. 16, 2021).

[18] J. E. Grayson, «Python and Tkinter Programming. Greenwich: Manning Publications Co.», p. 11, 2000.

[19] R. Pandey, P. Pidlypenskyi, S. Yang, i C. Kaeser-Chen, «Egocentric 6-DoF Tracking of Small Handheld Objects», abr. 2018, Accedit: ago. 15, 2021. [En línia]. Disponible a: http://arxiv.org/abs/1804.05870.

[20] «Wansview Cámara Vigilancia WiFi Exterior, 1080P Cámara IP WiFi de Seguridad con Visión Noturna Detección de Movimiento Audio Bidireccional, Soporta Alexa RTSP Onvif, IP66 Impermeable, W6 (Blanco) : Amazon.es: Electrónica». https://www.amazon.es/dp/B08CXDZMKH?tag=camarasvigiauto-21&linkCode=osi&th=1&psc=1 (accedit ago. 05, 2021).

[21] «Cámara Digital Nikon D3200». https://www.nikon.es/es_ES/product/discontinued/digital-cameras/2016/d3200#tech_specs (accedit ago. 15, 2021).

[22] «Webcam Logitech 4K Pro con HDR y RightLight 3». https://www.logitech.com/es-es/products/webcams/brio-stream-4k-hd-webcam.960-001194.html?crid=34 (accedit ago. 15, 2021).

[23] «Cable USB Tipo C, RAVIAD Cable USB C a USB 3.0 Cable Tipo C Carga Rápida y Sincronización Compatible con Galaxy S10/S9/S8/Note 10, Huawei P30/P20, Mi A1/Mi A2 y más - 1M, Gris : Amazon.es: Informática». https://www.amazon.es/RAVIAD-Rápida-Sincronización-Compatible-Galaxy/dp/B089GD78ZW/ref=sr_1_5?__mk_es_ES=ÅMÅŽÕÑ&dchild=1&keywords=cable+usb-c+a+usb&qid=1629045925&sr=8-5 (accedit ago. 15, 2021).

[24] «Rankie Cable Mini HDMI a HDMI, vídeo 4K, Compatible con Ethernet, 3D y ARC, 1,8 m, Negro : Amazon.es: Electrónica». https://www.amazon.es/Rankie-Cable-vídeo-Compatible-Ethernet/dp/B01KRKO4MM/ref=sr_1_5?adgrpid=59028305474&dchild=1&gclid=CjwKCAjw9uKIBhA8EiwAYPUS3PrmWbCGh_PDv5A4QutnjESIShg6bFfI-Oikk6xN91Qax4FdDQhuGhoCaTkQAvD_BwE&hvadid=275357092486&hvdev=c&hvlocphy=1005429&hvnetw=g&hvqmt=e&hvrand=7347689989545647613&hvtargid=kwd-300393310709&hydadcr=23390_1797049&keywords=cable+mini+hdmi&qid=1629041836&sr=8-5 (accedit ago. 15, 2021).

[25] «DIWUER Capturadora de Video HDMI, 4K HDMI a USB 2.0 Convertidor Video Audio, HDMI Vídeo Game Capture 1080P 30FPS para Edite Video/Juego/Transmisión/Enseñanza en línea : Amazon.es: Informática». https://www.amazon.es/DIWUER-Capturadora-Convertidor-Transmisión-Enseñanza/dp/B08BC52KCT/ref=sr_1_2_sspa?__mk_es_ES=ÅMÅŽÕÑ&crid=3C4AJFI07VRC0&dchild=1&keywords=capturadora+video+hdmi&qid=1629042214&sprefix=capturadora%2Caps%2C195&sr=8-2-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzVDRNNVREWVVJNUZLJmVuY3J5cHRlZElkPUEwMzc1NjUwMlA2SDAwSTBRWVhaVSZlbmNyeXB0ZWRBZElkPUEwMjc4MjM0M0ZVRDRDRlBRUjA5OSZ3aWRnZXROYW1lPXNwX2F0ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU= (accedit ago. 15, 2021).

9 Annexos

9.1 Guia d'instal·lació

A continuació es detallaran els passos a realitzar per poder instal·lar i executar l'aplicació amb els menors problemes possibles.

1. Descarregar i instal·lar Visual Studio Code [11].
2. Descarregar el zip amb l'aplicació.
3. Obrir el Visual Studio Code i obrir el directori de l'aplicació.
4. Instal·lar les llibreries necessàries amb l'ajuda de les següents comandes:
    a. pip install opencv-python
    b. pip install dlib
    c. pip install pillow
    d. pip install imutils
5. Executar l'aplicació amb la següent comanda (tenir en compte el directori d'execució):
    a. py pplCapacityCalculator.py