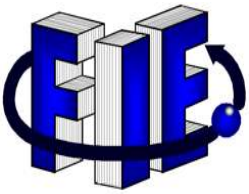

Universidad Michoacana de San Nicolás de Hidalgo

Facultad de Ingeniería Eléctrica

SISTEMA DE VISIÓN COMPUTACIONAL PARA EL CONTEO DE PERSONAS, HEURÍSTICA PARA EL SEGUIMIENTO DE OBJETOS

TESIS

Que para obtener el grado de
INGENIERO EN COMPUTACIÓN

Presenta
Misael Armenta Nieto

Asesor de Tesis
M.I. Moisés García Villanueva

Morelia, Michoacán, Diciembre 2015

Agradecimientos

Quiero agradecer a la Facultad de Ingeniería Eléctrica de la Universidad Michoacana de San Nicolás de Hidalgo por acogerme en sus aulas donde recibí la formación académica por parte de los profesores, a los cuales también les agradezco por dicha enseñanza en especial al profesor Moisés García Villanueva quien siempre fue un excelente instructor apoyándome en muchos de los proyectos que me propuse, también le agradezco su dirección y apoyo en la elaboración de esta tesis.

Agradezco a mi familia, amigos y a todas las personas que de alguna u otra manera siempre fueron una inspiración para que yo siguiera adelante con mi formación, dándome una palabra de aliento o apoyo, todos ellos fueron de gran ayuda para mi, pero principalmente mi madre quien siempre me apoyó en todo momento sin dudarlo un solo momento brindándome lo que yo necesité todo el tiempo.

A mis hermanos les agradezco el darme su apoyo incondicional y el estar en todo momento conmigo, levantándome el ánimo cada vez que yo lo necesite.

Finalmente quiero agradecer a dios por darme salud, buenos amigos, y una gran familia a la cual quiero mucho, por darme la sabiduría suficiente y haberme guiado en todo este camino dándome la fuerza suficiente para lograrlo.

Dedicatoria

Esta tesis la dedico a mis padres quienes me dieron la vida, pero principalmente a mi mamá Gloria Nieto Ramírez por brindarme cariño y apoyo incondicional tanto moral como económico en todo momento, aconsejándome y guiándome en mi trayecto estudiantil y de vida, se la dedico a ella por el sacrificio y esfuerzo que hizo en darme una carrera profesional. A mis hermanos José Rafael Armenta Nieto, Yuliana Armenta Nieto y Daniela Morraz Morales que siempre han estado presentes en todo este camino que he tenido que recorrer, siendo un gran apoyo para mi el saber que me apoyan incondicionalmente y dándome esos ánimos que necesité en todo momento.

A mis amigos que siempre estuvieron motivándome con una palabra de aliento, o una charla y brindándome su amistad.

A mis profesores por su tiempo dedicado, por su apoyo brindado, así como por la sabiduría que me transmitieron mediante el desarrollo de mi formación profesional, en especial al profesor Moisés García Villanueva quien aún fuera de las aulas y horarios de clase seguía instruyéndome, y por haber compartido conmigo logros obtenidos.

Dedico esta tesis a dios quien siempre me ha estado guiando por el mejor camino posible y cuidando en todo momento.

Resumen

Desde tiempos muy antiguos siempre ha sido indispensable tener un control de acceso de personas (conteo de personas), por ejemplo: en empresas, fábricas, escuelas. Siempre es necesario tener un control de acceso para sus trabajadores, de tal forma que se logre identificar a las personas que no pertenecen al personal. También es necesario que se pueda controlar la hora de entrada y salida de los mismos trabajadores en los sitios que laboran. Este tipo de control se requiere en casi todo lugar donde exista un gran flujo de personas, como lo pueden ser casi cualquier tipo de eventos como los de música, arte, deporte, cultura, entre muchos otros.

En la actualidad, para lograr tener el control de acceso de personas (conteo de personas), se han desarrollado muchos sistemas y dispositivos que ayudan a realizar esta tarea, empleando equipos mecánicos, electrónicos y sofisticadas aplicaciones de software. Algunos de estos desarrollos son: torniquetes, puertas giratorias, contadores manuales, electrónicos, térmicos y por medio de software.

En este trabajo se presentan dos heurísticas nuevas para el seguimiento de objetos, basadas en los contornos, mostrando su aplicación en el diseño e implementación de un sistema para el conteo de persona por medio de visión computacional. Se describe el procedimiento de análisis sobre un vídeo para identificar personas que acceden a un espacio físico y con dicha información realizar el conteo de personas que entran y salen. Se proyecta la colocación del sistema en la parte superior del acceso físico. El proceso para lograr el objetivo planteado inicia con identificar el fondo de la imagen, para ello se captura una primera imagen (denominada fondo) la cual se transforma a escala de grises, se le aplica un filtro de umbralización para eliminar cierta información no útil y se obtiene la diferencia con cada imagen del vídeo, obteniendo las regiones de interés que contienen los objetos. Posteriormente mediante la técnica de identificación de contornos se obtiene el área para cada uno de ellos y se define un valor de umbral para filtrarlos, entonces se ajustan los puntos del contorno a un rectángulo al cual se le da seguimiento en las subsecuentes imágenes de vídeo, generando una trayectoria del movimiento del objeto identificado. Finalmente con la trayectoria se identifica la dirección del movimiento de la persona (entrada o salida del acceso físico) y se cuenta el número de trayectorias que cruzan por un punto determinado en el sistema, lo que equivale al número de personas identificadas. El sistema de seguimiento presentado es del tipo un sólo objeto a la vez.

Palabras clave: visión computacional, identificación de objetos, conteo de personas, algoritmos, seguimiento de objetos, heurísticas.

Abstract

From very ancient times it has always been essential to have an access control people (in our contex counting people), for example: phisical spaces in companies, factories and schools. It is always necessary to have an access control for their workers or others people, in the case to identify people who do not belong to the staff, it is very necessary to be able to control the time of entry and exit of these people on sites that they access or work, this type of control is required in almost every place where there is a great flow of people, some of which we are appointed are any type of events such as music, art, sports, culture, among others.

Until today to achieve access control people (people counting), many systems that have been developed and devices that help perform this task are using mechanical systems, electronic devices, and sophisticated software applications. Some of these developments are turnstiles, revolving doors, manual counters, electronics, thermals and counters through software.

This work presents two heuristics for tracking objects, based on contours, showing its application in the design and implementation of a system for the counting of people using computer vision. The analysis procedure on a video to identify people accessing a physical space and that information make people counting incoming and outgoing described. Placing the system on top of physical access projects. The process for achieving the objective starts with identifying the background image, getting the first frame (called background), which is converted to grayscale and filtered by a thresholding filter to remove some useless information, then we get next frames of the video and compute the difference, so with the image computed we are getting regions of interest that contain objects. The system proceeed using the heuristic that identify contours where each one is filtered by a threshold value defined (countour's area), a candidate contour or set of contours are represented by one rectangle that cover all them, that rectangle is followed up in the next video frames catches, generating a path of movement of the identified object. Finally the path that describe the direction of movement of the person (or out of physical access) is identified and the number of paths that cross a certain line in the system, equivalent to the number of people identified is counted. The system presented is a single object tracking.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Símbolos

CV	**C**omputer **V**ision (Visión computacional)
fps	**f**rames **p**er **p**econd (Cuadros o imágenes por segundo)
ROI	**R**egion **O**f **I**nterest (Región de intéres)
IEEE	**I**nstitute of **E**lectrical and **E**lectronics Engineers (Instituto de Ingeniería Eléctrica y Electrónica)
CVPR	**C**omputer **V**ision **P**attern Recognition ()
BSD	**B**erkeley **S**oftware **D**istribution (Distribución de Software Berkeley)
HCI	**H**uman **C**omputer **I**nteraction (Interacción Humano Computadora)
SFM	**S**tructure **F**rom **M**otion (Estructura Obtenida del Movimiento)
ML	**M**achine **L**earning (Aprendizaje de Máquina)
GUI	**G**raphical **U**ser **I**nterface (Interfaz Gráfica de Usuario)
NSF	**N**ational **S**cience **F**oundation (Fundación Nacional de Ciencia)
3D	**3 D**imention (Tridimensional)
CPU	**C**entral **P**rocessing **U**nit (Unidad Central de Procesos)
RAM	**R**andom **A**ccess **M**emory (Memoria de Acceso Aleatorio)
SD	**S**tandard **D**efinition (Definición Estándar)
PAL	**P**hase **A**lternating **L**ine (Línea de Fase Alternada)
NTSC	**N**ational **T**elevision **S**ystem **C**ommittee (Comisión Nacional de Sistema de Televisión)

Capítulo 1

Introducción

1.1. Identificación del problema

Entre 1980 y 2000 la población mundial total aumentó de 4,400 millones a 6,000 millones. Según las proyecciones, se calcula que para el año 2015 habrá aumentado por lo menos en otros 1.000 millones, lo que hará un total de más que 7,000 millones [Ban15].

El crecimiento poblacional en espacios cada vez más reducidos, genera diversos problemas de índole social y económica, entre los que se destacan, la inseguridad, las dificultades en la movilidad y la contaminación ambiental. Algunos de estos problemas se evidencian con más fuerza en determinados sitios de la ciudad, por ejemplo, en zonas comerciales, administrativas y educativas, ya que en éstas se presenta alta afluencia de personas [Cortés S.13].

Las autoridades gubernamentales y privadas, con jurisdicción en zonas de alta afluencia, deben velar por la comodidad y seguridad de la gente, por tal razón, han desarrollado planes de ordenamiento, que han implicado grandes sumas de dinero. A pesar de ello, las inversiones han sido insuficientes, debido a falta y deficiencia de estudios que revelen el comportamiento, las necesidades y algún estimado de la cantidad de transeúntes en ciertos lugares [Cortés S.13].

En muchas ocasiones es necesario saber cuántas personas cruzan algún acceso ya sea para entrar o salir de algún lugar, como lo son centros comerciales, escuelas, aulas

o cualquier otro lugar donde exista una gran concurrencia de personas. En los centros comerciales es muy deseable saber qué días y a qué hora es cuando existe la mayor cantidad o menor cantidad de clientes, de esta forma ellos pueden planear sus estrategias de mercado, y así saber en qué momentos es necesario contar con mayor o menor número de trabajadores y en qué áreas son las que se requieren dichos trabajadores.

En muchos lugares es también muy deseable contar el número de personas que se presentan a algún evento masivo, y de esta forma se le puede dar una mejor estrategia de seguridad, de esta manera poder solucionar o evitar problemas que se puedan presentar antes, después o durante el evento; también puede ser útil saber el momento en que se llegue a la capacidad máxima y poder cumplir con las normas de seguridad, así como optimizar los recursos disponibles, controlando por ejemplo la ventilación, calefacción y sistemas de aire acondicionado disponibles [Cortés S.13].

En la actualidad existen una gran cantidad de equipos, dispositivos y aplicaciones para lograr el mejorar la seguridad desde la detección de objetos, hasta el seguimiento o rastreos de alguna parte en específico para su posible identificación, como lo puede ser la identificación de rostros o personas, identificación de placas en los vehículos, o simplemente saber a qué distancia se encuentra algo desde un punto en específico. Lo más deseable para un sistema de seguridad o de seguimiento de objetos, es poder realizar de manera simultánea el seguimiento en todas las escalas [Masoud01].

Los contadores de personas en la actualidad son una gran herramienta útil para distintas aplicaciones. Todas esas aplicaciones tienen el mismo fin en común: conocer el número total de presentes en algún lugar en específico, por lo que también es necesario saber la dirección en la que transitan [Cuevas13].

El trabajo desarrollado en esta tesis tiene como fin la creación de un sistema de conteo de personas usando visión computacional. La idea es que el sistema pueda ser utilizado en espacios físicos donde existan accesos para entrar o salir como en escuelas, aulas, centros comerciales, lugares públicos, estadios, auditorios o en alguna empresa; detectar la dirección en la que los peatones transitan y la hora en que lo hacen.

1.2. Antecedentes

Existen diferentes aplicaciones comerciales para el conteo de personas, los cuales aún tienen desventajas grandes frente al costoso software especializado para el conteo de personas, dado que requiere de dispositivos costosos y sus aplicaciones son limitadas [Cuevas13].

Existen diversos métodos para el conteo de personas, sin utilizar un software costoso, tal es el caso del contador mecánico que consiste en torniquetes y puertas giratorias colocados en las entradas de las instalaciones. Sin embargo, este sistema tiene el problema de no permitir el flujo rápido en las entradas cuando existe mucha afluencia, ver la figura 1.1. Otro caso es el contador térmico, el cual tiene un sensor que identifica el gradiente entre la temperatura ambiental y la temperatura corporal. Sin embargo, este tipo de contador es impreciso cuando la temperatura ambiental es semejante a la corporal. El contador manual permite que una persona detecte el flujo peatonal de manera manual, con la evidente desventaja que una persona tiene cuando existen grandes acumulaciones de personas, además de que el contador manual usualmente tiene sólo 4 dígitos [Cuevas13] ver la figura 1.2.

Figura 1.1: Dispositivo mecánico para el conteo de personas, con flujo lento de personas.

Figura 1.2: Dispositivo manual para el conteo de personas, señalando que cuenta cuenta únicamente con 4 dígitos.

Otra técnica utilizada es el contador mediante rayos infrarrojos, ver la figura 1.3, es decir, un dispositivo infrarrojo transmisor se comunica con un dispositivo receptor (situados en lados opuestos de un pasillo, por ejemplo), y cuando una persona pasa, la comunicación se interfiere y el receptor agrega una persona a la cuenta. Este dispositivo es incapaz de detectar si dos o mas personas pasan al mismo tiempo (problema de oclusión). Es decir, al cruzar un grupo de personas al mismo tiempo, el contador sólo detecta una [Cuevas13].

Figura 1.3: Dispositivo electrónico para el conteo de personas, el cual utiliza de emisión de rayos de luz infrarroja.

Hoy en día el conteo de personas mediante el uso de videocámaras ha tomado un gran impulso en el ámbito de la investigación. El conteo de personas es un tema realmente

redituable pues el software y hardware comercial para este fin es realmente costoso, por lo que generar un nuevo mecanismo a bajo costo es un tema de interés [Cuevas13].

Actualmente existen estudios acerca del conteo de personas mediante videocámaras. Sin lugar a dudas, con el auge que ha tomado la tecnología en cuanto a imágenes y vídeo, ha sido posible la adquisición de cámaras a muy bajo costo, el cual permite tener un almacenamiento confiable de los sucesos acontecidos durante el día con lo que no sólo puede ser utilizado con fines de seguridad, sino también para el conteo de personas [Cuevas13].

En este trabajo se implementa un sistema de visión computacional utilizando la biblioteca OpenCV (del Inglés Open Source Computer Vision Library) para el conteo de personas; para identificar un objeto en movimiento se emplea la identificación del fondo de la imagen, se realiza la segmentación y seguimiento de los objetos que se encuentran desplazándose en la escena, se identifican las personas y la trayectoria que realiza, finalmente se obtiene el conteo de las personas.

1.3. Objetivo

Este trabajo tiene como objetivo el desarrollar un sistema para el conteo de personas mediante el uso de técnicas de visión computacional, abordar la problemática de seguimiento de objetos un objeto a la vez, emplear una sola cámara de bajo costo considerando lugares con distintos niveles de iluminación y afluencia peatonal.

1.4. Justificación

Existe una gran necesidad de implementar desarrollos tecnológicos en nuestro entorno local en el área de seguridad y otras áreas de aplicación de los sistemas de visión computacional, además se requiere de profesionales capacitados en el manejo e implementación de técnicas de visión computacional.

1.4.1. Objetivos particulares

- Adquisición de video en tiempo real.

- Detectar el fondo en cada una de las imágenes del video.
- Segmentar las imágenes para identificar objetos desplazándose en la escena.
- Seguimiento de objetos detectados, un sólo objeto a la vez.
- Detectar personas en la escena.
- Detectar la trayectoria de movimiento de las personas.
- Contar las personas en el sentido de afluencia.
- Realizar las pruebas y análisis de los resultados.

1.5. Descripción de los capítulos

- En el capítulo 1 se da una breve introducción a este trabajo. Se mencionan antecedentes, se plantea el problema y se establecen los objetivos principales y particulares de la tesis. Se señala además una descripción de cada capítulo.
- En el capítulo 2 se trata del área de visión computacional para el conteo de personas, y las bibliotecas de software existentes, mencionando además las técnicas utilizadas.
- En el capítulo 3 se trata del seguimiento de objetos en un sistema de visión computacional y la problemática en el sistema de identificación de personas.
- En el capitulo 4 se plantea el sistema propuesto en este trabajo, su diseño e implementación.
- En el capítulo 5 se realiza la parte experimental, las pruebas con el conjunto de datos sintéticos y con video en tiempo real capturado en distintos espacios. Se presentan además los resultados obtenidos.
- En el capítulo 6 se presentan las conclusiones de los resultados obtenidos, así como aportaciones y trabajos futuros.

Capítulo 2

Visión computacional y el conteo de personas

2.1. ¿Qué es visión?

Visión es la ventana al mundo de muchos organismos. Su función principal es reconocer y localizar objetos en el ambiente mediante el procesamiento de las imágenes. La visión computacional es el estudio de estos procesos, para entenderlos y construir máquinas con capacidades similares. Existen varias definiciones de visión, entre éstas podemos mencionar las siguientes [Sucar11]:

- "Visión es saber que hay y dónde mediante la vista" (Aristóteles).
- "Visión es recuperar de la información de los sentidos (vista) propiedades válidas del mundo exterior", [Gibson14].
- "Visión es un proceso que produce a partir de las imágenes del mundo exterior una descripción que es útil para el observador y que no tiene información irrelevante", [Marr82a]

Las tres son esencialmente válidas, pero la que tal vez se acerca más a la idea actual sobre visión computacional es la definición de Marr [Marr82a]. En esta definición hay tres

aspectos importantes que hay que tener presentes: (1) visión es un proceso computacional, (2) la descripción a obtener depende del observador y (3) es necesario eliminar la información que no sea útil (reducción de información) [Sucar11].

2.2. Visión computacional

Una área muy ligada a la de visión computacional es la de procesamiento de imágenes. Aunque ambos campos tienen mucho en común, el objetivo final es diferente. El objetivo de procesamiento de imágenes es mejorar la calidad de las imágenes para su posterior utilización o interpretación, por ejemplo [Sucar11]:

- Remover defectos
- Remover problemas por movimiento o desenfoque
- Mejorar ciertas propiedades como color, contraste, estructura, etc.
- Agregar "colores falsos" a imágenes monocromáticas.

En la figura 2.1 se ilustra el enfoque de procesamiento de imágenes, en el cual se obtiene una imagen "mejor" para su posterior interpretación por una persona [Sucar11].

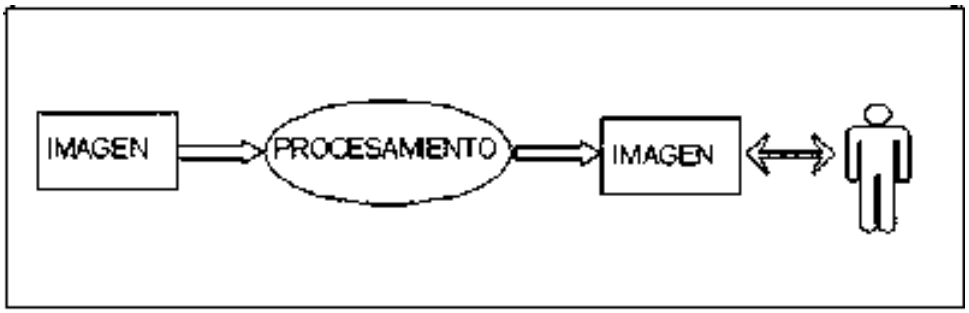

Figura 2.1: Esquema general del procesamiento de imágenes. Su función principal es presentar la misma imagen resaltando e ignorando ciertas características. Obsérvese que la entrada y salida son imágenes.

Basándose en la descripción de [Marr82b] sobre lo que es visión, se puede decir que visión por computadora es el proceso de captura, análisis, identificación de patrones útiles de una imagen digital para interpretarla y ejecutar alguna acción. Los sistemas de visión

por computadora realizan tareas de inspección con un alto nivel de flexibilidad y repetición, nunca se cansan o distraen y pueden trabajar en ambientes donde un inspector humano no podría realizar la inspección visual o le costaría mucho trabajo realizarla.

2.2.1. Objetivo de la visión computacional

El objetivo de la visión computacional es extraer características de una imagen para su descripción e interpretación por la computadora. Por ejemplo:

- Determinar la localización y tipo de objetos en la imagen.
- Construir una representación tridimensional de un objeto.
- Analizar un objeto para determinar su calidad.
- Descomponer una imagen u objeto en diferentes partes.

En visión se busca obtener descripciones útiles para cada tarea a realizar. La tarea demandará modificar ciertos atributos. La figura 2.2 muestra un esquema general de la visión por computadora [Sucar11].

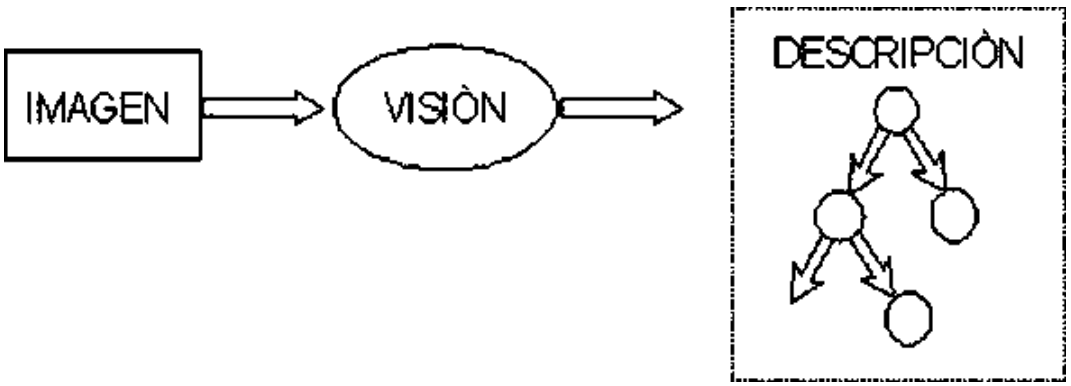

Figura 2.2: Esquema general de visión por computadora. La imagen de entrada es procesada para extraer los atributos, obteniendo como salida una descripción de la imagen analizada.

En la figura 2.3 se muestra un ejemplo de procesamiento de imágenes. La tarea a realizar es mejorar la imagen de entrada, la cual es obscura. La imagen de salida es esencialmente la misma pero de mejor calidad o "más útil". La figura 2.4 ilustra la diferencia

entre procesamiento de imágenes y visión; notese que la imagen muestra ciertas descripciones importantes, como los números, que previamente fueron detectados. La salida de este sistema de visión se complementa con un módulo de reconocimiento de patrones, es decir, "saber" que letras y números contiene la placa [Sucar11].

Figura 2.3: Aumento de contraste: (a) imagen obscura debido a que su rango de grises es reducido, (b) ecualización del rango de grises.

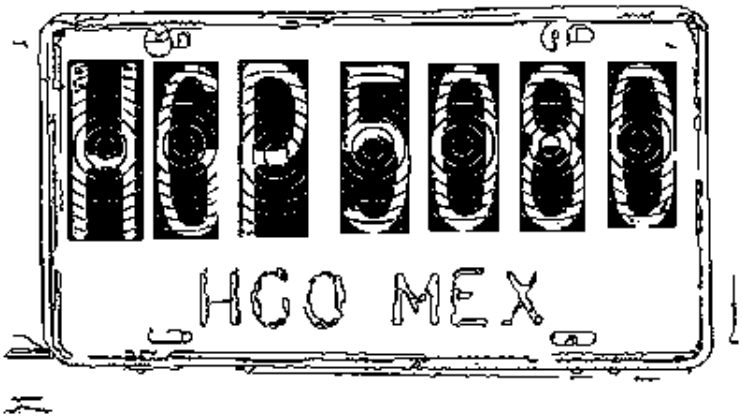

Figura 2.4: Reconocimiento de caracteres en base a su codificación radial.

Actualmente existen múltiples aplicaciones prácticas de la visión computacional, entre éstas podemos mencionar las siguientes [Sucar11]:

- Robótica móvil y vehículos autónomos. Se utilizan cámaras y otros tipos de sensores

para localizar obstáculos, identificar objetos y personas, encontrar el camino, etc.

- Manufactura. Se aplica visión para la localización e identificación de piezas, para control de calidad, entre otras tareas.

- Interpretación de imágenes aéreas y de satélite. Se usa procesamiento de imágenes y visión para mejorar las imágenes obtenidas, para identificar diferentes tipos de cultivos, para ayudar en la predicción del clima, etc.

- Análisis e interpretación de imágenes médicas. La visión se aplica para ayudar en la interpretación de diferentes clases de imágenes médicas como rayos-X, tomografía, ultrasonido, resonancia magnética y endoscopia.

- Análisis de imágenes microscópicas. El procesamiento de imágenes y visión se utilizan para ayudar a interpretar imágenes microscópicas en química, física y biología.

- Análisis de imágenes para astronomía. Se usa la visión para procesar imágenes obtenidas por telescopios, ayudando a la localización e identificación de objetos en el espacio.

- Análisis de imágenes para compresión. Aunque la compresión de imágenes ha sido tradicionalmente una subárea del procesamiento de imágenes, recientemente se están desarrollado técnicas más sofisticadas de compresión que se basan en la interpretación de las imágenes.

2.2.2. Ventajas en visión computacional

La visión por computadora tiene ventajas en tareas que se realizan a velocidades muy altas y en las tareas que requieren estar repitiéndose constantemente, por ejemplo en la inspección de tareas o procesos de ensamblaje y que se trabaje sin interrupciones. Existen tareas o procesos donde se requiere inspección visual. Los sistemas de visión tienen una clara ventaja sobre la visión humana, ya que los margenes de error son muy reducidos. Se evitan algunos factores como el cansancio ya sea visual, muscular o distracciones, además de que disminuye el personal para realizar las inspecciones y/o revisiones por lo que se tiene

un costo menor y es más preciso a la hora de realizar una inspección con un sistema de visión artificial en comparación con los inspectores humanos. Otra ventaja de un sistema de visión por computadora es que se pueden realizar tareas con un horario amplio, es decir, se pueden realizar tareas durante largos periodos de tiempo sin que esto requiera un costo muy elevado debido al pago de horas extras al personal [Reyes15].

2.2.3. Desventajas en visión computacional

Una de las desventajas de la visión computacional es la iluminación que se puede presentar en los diferentes escenarios donde se desempeñan los sistemas de visión computacional. En comparación con la visión humana (los ojos y el cerebro), los sistemas de visión computacional son muy básicos, además de que en un humano la velocidad de interpretación es muy alta en comparación con un sistema de visión computacional. Otro problema que se presenta en un sistema de visión computacional son los problemas de colocación o posición de las cámaras, algunos sistemas deben contar con cierto rango de visión para poder ejecutar de manera correcta las tareas que debe realizar [Reyes15].

2.2.4. Componentes de un sistema de visión por computadora

Los principales componentes de un sistema de visión por computadora son: Una cámara de video o algún objeto similar para captura de imágenes y el software necesario para desarrollar la aplicación. La función de las cámaras de video es capturar la imagen proyectada para mandarla a un sistema electrónico para que pueda ser interpretada, almacenada y/o simplemente visualizada. El sistema electrónico puede ser una computadora para visualizar, almacenar y procesar la imagen. El software es un elemento muy importante en una aplicación de visión computacional ya que es el encargado de analizar, procesar e identificar las características de las imágenes capturadas [Reyes15].

2.3. Video

Para el desarrollo de un sistema de visión computacional para alguna tarea específica se utiliza la grabación de video o captura de video en tiempo real, el cual debe ser

analizado o procesado por el programa que realizará el análisis en el sistema, por lo cual es necesario dar una explicación de que es y en que consiste el video (grabación o captura de video).

El video es un sistema de grabación y reproducción de imágenes, consiste en la captura de una serie de fotografías (en este contexto llamadas "fotogramas") que luego se muestran en secuencia y a gran velocidad para reproducir la escena original, un fotograma o frame es cada una de las imágenes que forman un video. Se expresan con las siglas fps (por sus siglas en Inglés de frame per second). En la Figura 2.5 se muestran una serie de frames que conforman constituyen un video.

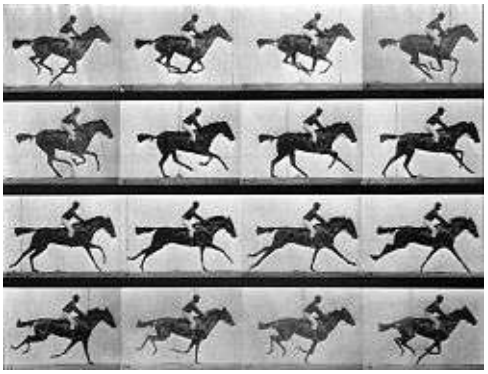

Figura 2.5: Fotogramas de un video mostrando un caballo en movimiento [Muy15]. Famosa serie de fotografías de Muybridge 1878.

La expresión fps se utiliza en el contexto del vídeo, el cine o gráficos por computadora. Se puede hacer una grabación en diferentes fotogramas por segundo, dependiendo del tipo de vídeo que sea, lo que vayamos a grabar o a dónde se va a reproducir [fra15].

Progresivo Vs entrelazado

Una imagen en movimiento puede formarse de diferentes maneras en función de si utilizamos los fotogramas de manera progresiva o entrelazada, la diferencia entre estos es mostrada en la Figura 2.6. Lo cual consiste en: [fra15].

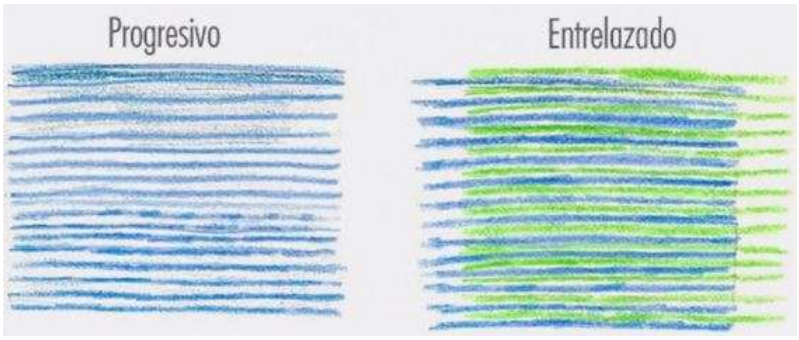

Figura 2.6: Diferencia entre progresivo contra entrelazado.

- En progresivo cada fotograma muestra todas las líneas que forman la imagen. El formato progresivo se expresa con una letra p, por ejemplo (1080p).

- En entrelazado cada fotograma sólo muestra la mitad de las líneas de una imagen, por lo tanto se necesitan dos fotogramas para mostrar la imagen completa. En imágenes estáticas la calidad está bien, pero si las imágenes contienen un movimiento rápido producen unas líneas como si fueran persianas. El formato entrelazado se expresa con la letra i, por ejemplo (1080i).

2.3.1. ¿Cómo era antes la velocidad del video?

En los comienzos del cine no había una velocidad de proyección base porque, al utilizar cámaras con manivela manual, era imposible mantener la misma velocidad todo el tiempo. Fue con la llegada del sonido sincronizado cuando se establecieron los 24 fotogramas por segundo como velocidad estándar.

El taumátropo, en inglés wonderturner, es uno de los juguetes precursores al cine. Es un disco con dos imágenes diferentes en ambos lados y un trozo de cuerda a cada lado del disco, cuando giramos los discos rápidamente con la cuerda produce la ilusión de que ambas imágenes están juntas [fra15], un ejemplo se muestra en la Figura 2.7.

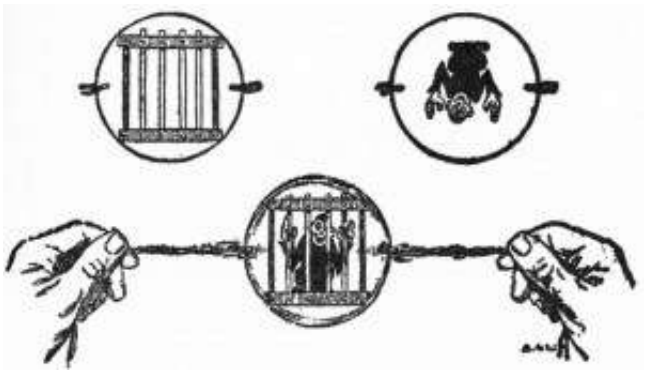

Figura 2.7: Ejemplo de traumátropo [ori10].

2.3.2. ¿Cómo es ahora la velocidad del video?

Vamos con los tipos de frame rate que existen ahora, y cómo y cuándo utilizarlos. Para simplificar, nombraremos solo los más conocidos que serían 24 fps en cine, 25 fps en sistema PAL de la televisión europea y 30 fps para el sistema NTSC de la televisión estadounidense. A partir de los 60 fps, como por ejemplo pueden ser los 100 fps o más, se utilizan para grabar cámaras lentas con una nitidez incomparable con las cámaras lentas de 24 fps o 25 fps. Cuántos más fotogramas compongan nuestra imagen, más fluida y nítida será la imagen mientras que la cámara será lenta. Esto puede ser muy útil para grabar deportes o movimientos muy rápidos, en La Figura 2.8 se muestran algunas imágenes capturadas a diferentes resoluciones [fra15].

2.4. Biblioteca OpenCV

OpenCV (Open Source Computer Vision) es una biblioteca o librería de funciones para visión por computadora, de código abierto, que está escrita en lenguajes C y C++. Fue diseñada para obtener alta eficiencia computacional enfocándose en aplicaciones en tiempo real. La librería cuenta con más de 500 funciones que abarcan muchas áreas en visión por computadora, incluyendo, manufactura, inspección de productos, imágenes médicas, seguridad, interfaz para el usuario, calibración de cámara, visión estéreo y robótica, entre otras. Además incluye una librería de aprendizaje de máquina de propósito general [Cortés S.13].

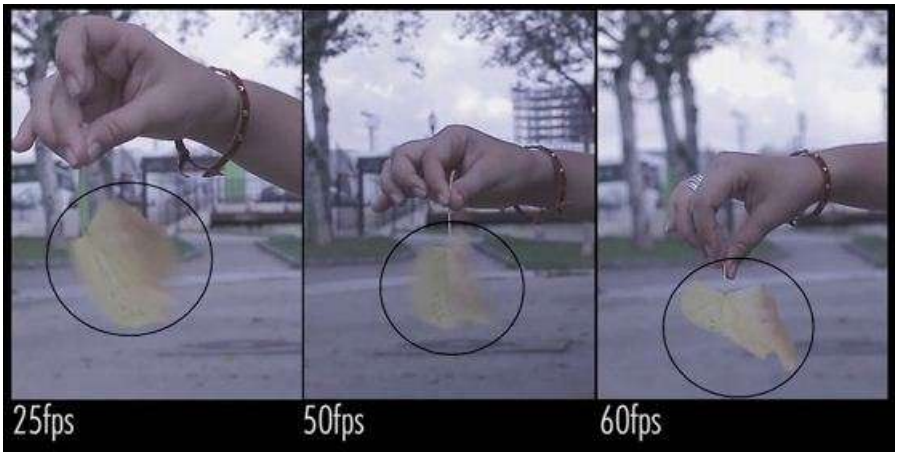

Figura 2.8: Imágenes capturadas a velocidades de 25, 50 y 60 fps donde se logra captar la nitidez de cada resolución.

2.4.1. Historia y datos sobresalientes sobre OpenCV

El 13 de Junio del 2000, Intel® Corporation anunció que estaba trabajando con un grupo de reconocidos investigadores en visión por computadora para realizar una nueva librería de estructuras/funciones en lenguaje C. Esta librería proporcionaría un marco de trabajo de nivel medio-alto que ayudaría al personal docente e investigador a desarrollar nuevas formas de interactuar con las computadoras. Este anuncio tuvo lugar en la apertura del IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). Había nacido The Open Computer Vision Library [Ope15] y lo hacía bajo licencia BSD (Software Libre) [Arévalo04].

2.4.2. Estructura y características de la librería OpenCV

La librería OpenCV esta dirigida fundamentalmente a la visión por computadora en tiempo real. Entre sus muchas áreas de aplicación destacan: interacción hombre-máquina (HCI); segmentación y reconocimiento de objetos; reconocimiento de gestos; seguimiento del movimiento; estructura a partir del movimiento (SFM); y robots móviles [Arévalo04]. OpenCV se estructura en cinco componentes principales, cuatro de los cuales se muestran y describen en la Figura 2.9 [Cortés S.13].

OpenCV contiene funciones para el procesamiento básico de imágenes y los algorit-

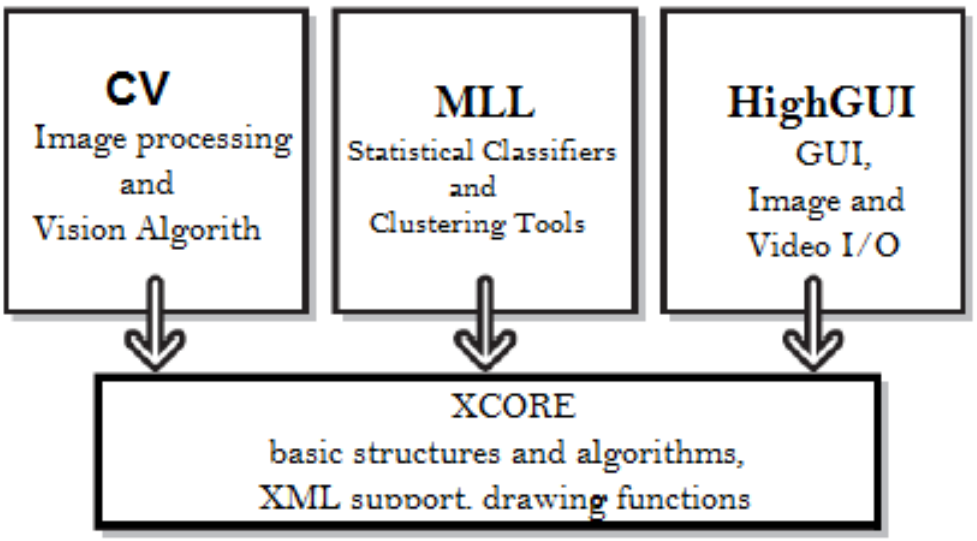

Figura 2.9: Estructura de la librería OpenCV [Cortés S.13].

mos de visión por computadora de alto nivel; ML es la librería de aprendizaje de máquina, que incluye, varios clasificadores estadísticos y herramientas de agrupamiento; HighGUI contiene rutinas Entrada/Salida y funciones para almacenar y cargar videos e imágenes y CXCore contiene las estructuras de datos y algoritmos básicos. Otro componente es el CvAux, que contiene áreas extintas (Reconocimiento Fácial) y algoritmos experimentales (Segmentación de Fondo y Primer Plano) [Bradski08].

OpenCV fue diseñado para ser de plataforma múltiple, ya que tiene versiones para GNU/Linux, Mac OS X y Windows.

Al ser OpenCV software libre, se da a los usuarios la libertad para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De una forma más precisa, los usuarios del software tienen cuatro libertades [HIS15]:

- La libertad de usar el programa con cualquier propósito (libertad 0).
- La libertad de estudiar cómo funciona el programa y adaptarlo a sus necesidades (libertad 1). El acceso al código fuente es una condición previa para esto.
- La libertad de distribuir copias (libertad 2).

- La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie. (libertad 3). El acceso al código fuente es un requisito previo para esto.

Intel facilitó el desarrollo y mantenimiento de la librería OpenCV aceptando y manteniendo las propuestas de código fuente revisadas por un grupo de expertos de la comunidad investigadora. Revisores que incluían representantes de los más importantes laboratorios de visión incluyendo: Jitendra Malik, University of California Berkeley; Takeo Kanade, Carnegie Mellon University; Pietro Perona, NSF Engineering Research Center, California Institute of Technology; Irfan Essa, Georgia Institute of Technology; Carlo Tomasi, Stanford University; Trevor Darrell, Massachusetts Institute of Technology; Stan Sclaroff, Boston University.

Son muchos los proyectos de Software Libre en avanzado estado de gestación que hacen uso de la librería OpenCV como marco de trabajo, entre ellos podemos destacar: Foresight; OpenCV User Contribution Library y S2iLib [Arévalo04].

2.5. Conteo de personas

El conteo de personas es un tema que ha cobrado gran importancia en los últimos años; se utiliza en muchos escenarios tales como: estaciones de metro, autobuses, tiendas, centros comerciales o cualquier escenario que requiera una contabilización de tránsito peatonal. Entre las razones más importantes para contar gente en un determinado escenario están [Mera G.15]:

- Inteligencia de Mercado que usan ciertos comercios para construir sus estrategias de Marketing, es decir que necesitan calcular el porcentaje de visitantes que hacen compras en una tienda (conversión de la tasa) y determinar el rendimiento que han tenido en ventas y la eficiencia de dicho Marketing.

- Optimización de turnos de personal, que mediante la densidad de tráfico (alta o baja) realizan un análisis para determinar cuándo ejecutar un determinado servicio; por ejemplo; un servicio de limpieza cuando la densidad de gente es baja.

Los primeros sistemas de conteo eran totalmente mecánicos y consistían en su mayoría de torniquetes que contaban con gran precisión, pero tenían problemas de eficiencia al trabajar bajo densidades de tráfico altas. Al aparecer nuevos escenarios (abiertos) como calles, parques y/o plazas, se evolucionó hacia sistemas de conteo que eviten la obstrucción del paso peatonal y que cuente con igual o mejor precisión que sus antecesores. Desde ese momento comenzaron a aparecer los primeros sistemas electrónicos de conteo, tales como sensores infrarrojos y cámaras térmicas, que envían señales electromagnéticas que son censadas por el mismo dispositivo para comprobar la presencia de alguien en una determinada zona. Con estos dispositivos se conseguía contar sin necesidad de colocar objetos físicos en medio del escenario que obstruyan el paso peatonal, pero eran excesivamente caros y tenían problemas para contar aglomeraciones de gente. A partir de esto, se comenzaron a desarrollar sistemas de Visión por Computadora que utilizan algoritmos computacionales para analizar las muestras de vídeo obtenidas a través de una (o varias) cámara(s) colocada(s) en la zona de interés y en donde se trata de resolver los problemas de oclusión mediante el procesado digital de imágenes [Mera G.15].

En los trabajos de [Kim02, Masoud01, Septian06, Yang03, Zhao02, Zhao09] que menciona [Cuevas13] puede observarse que la segmentación y detección de personas es un elemento primordial y resulta la primera parte a realizar por el algoritmo que se desea proponer, ya que una vez hecha la detección de personas es posible realizar el seguimiento y finalmente el conteo de esta persona.

Los trabajos realizados por [Kim02, Masoud01, Septian06, Yang03, Zhao02, Zhao09] son métodos de detección de personas, y se observó que todos ellos son mecanismos buenos, que tienen muchas ventajas y realmente pocas desventajas. Discernir entre que método es el mejor no es una tarea fácil. El criterio para tomar una elección no es simple pues existen muchas ventajas, por ejemplo, trabajar con simples siluetas, pero también existen ventajas en trabajar con el cuerpo humano completo o con el rostro de las personas. Estos criterios se eligen de acuerdo a la aplicación especifica que se le desea dar al sistema propuesto [Cuevas13].

2.6. Contadores de personas basados en sistemas de visión

Exiten muchos métodos para realizar el conteo de personas utilizando distintos tipos de cámaras como lo son:

- Detección de cabezas por estéreo visión.
- Detección de rostros.

2.6.1. Detección de cabezas por estéreo visión

Existen diferentes métodos para la detección de cabezas. Un método es realizar una búsqueda de la perspectiva circular de la cabeza de cada una de la personas, a través de un modelo bidimensional. Este método de detección no introduce errores en cuanto al número de personas representado por una región de interés [García12].

Para obtener una detección más robusta se realiza un filtrado de alturas por estéreo visión, al obtener el valor 3D de altura de la persona. En el trabajo [Patil04], se realiza un seguimiento de rostros donde las cámaras están situadas a la altura de las personas. Para ello, utilizan técnicas que incluyen características como color, forma, etc. incluyen el uso de un modelo con color tipo Ω para aumentar la eficiencia del sistema. Debido a la situación del sistema y las diferentes posibilidades de trayectorias, se propone un modelo más general de tipo circular para la detección. En este caso el seguimiento de personas se realiza en diferentes sentidos respecto la situación de la cámara, por lo que no es válido utilizar un modelo con color, ya que no se pueden contemplar todas las situaciones que pueden suceder con un mismo modelo con color. Por este motivo, únicamente se busca la característica circular de la cabeza [García12].

En el trabajo de [García12] se realizaron diferentes pruebas con dos modelos de diferentes anillos circulares. En la figura 2.10 inciso a) se presenta el modelo más básico, compuesto únicamente por un anillo, mientras que en la figura 2.10 inciso b) se presenta el modelo propuesto. Con este perfil que se propone, se asegura que la zona analizada tenga forma circular.

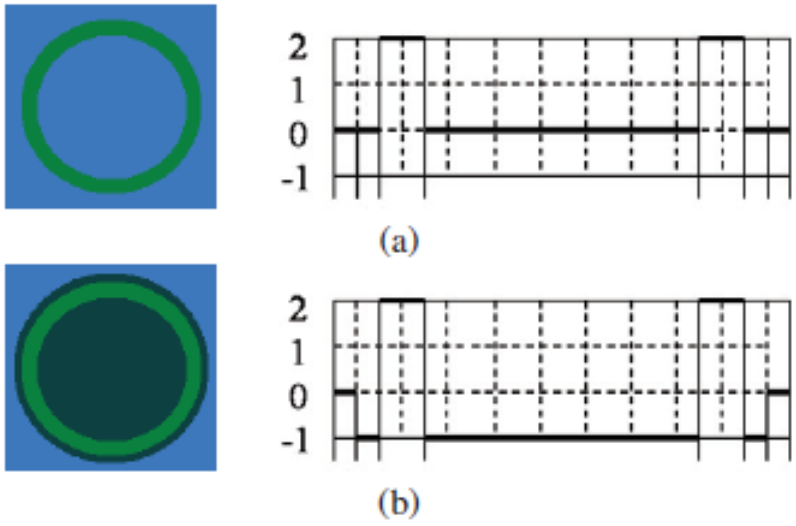

Figura 2.10: Máscaras de detección y valores del perfil: (a) Máscara básica. (b) Máscara propuesta [García12].

2.6.2. Detección de rostros

En los sistemas de conteo de personas que utilizan la detección de rostros, son sistemas diseñados para lugares cerrados con moderada concentración de personas con una cámara ubicada de manera estratégica, por lo que problemas como la falta o cambio drástico de condiciones naturales como luz, temperatura, viento no son problema, pues es un lugar estable. Este sistema resuelve el problema de la oclusión mediante el seguimiento de rostros y además resuelve el problema de decidir si las detecciones que aparecen en las imágenes necesariamente son personas. El conteo de personas se realiza mediante el análisis de las trayectorias de los peatones. Aunque el conteo es realizado de manera unidireccional, es decir, el sistema no puede realizar la detección de personas si la posición no es de frente a la cámara [Cuevas13].

Una vez establecida la manera en la cual se realizará la detección de personas en este caso mediante detección de rostros y el seguimiento de los mismos, es importante saber acerca de las distintas técnicas existentes para la detección de rostros. En [Hjelmås01], y [Yang02] pueden observarse distintos métodos existentes de detección de rostros.

En este caso, el sistema para el conteo de personas sera colocado en un ambiente estable con las siguientes condiciones:

- Temperatura ambiente.
- Iluminación continua (no muy variante).
- Poca posibilidad de viento fuerte (lugares cerrados como bancos o centros comerciales).

De cualquier forma es bueno considerar algunas variaciones en el ambiente en el cual se desenvuelve el sistema para el conteo de personas, el cual es un ambiente móvil con lo cual dejarían de cumplirse las condiciones mencionadas anteriormente.

Capítulo 3

Segmentación y seguimiento de objetos

3.1. Niveles de visión

Visión consiste en partir de una imagen (píxeles) y llegar a una descripción (predicados, geometría, formas, etcétera.) adecuada de acuerdo a nuestro propósito. Como este proceso es muy complejo, se ha dividido en varias etapas o niveles de visión, en cada una se va refinando y reduciendo la cantidad de información hasta llegar a la descripción deseada. Se consideran generalmente tres niveles [Sucar11]:

- **Procesamiento de nivel bajo** - se trabaja directamente con los píxeles para extraer propiedades como: orillas, gradiente, profundidad, textura, color, etc.

- **Procesamiento de nivel intermedio** - consiste generalmente en agrupar los elemento obtenidos en el nivel bajo, para obtener líneas, regiones, generalmente con el propósito de la segmentación.

- **Procesamiento de alto nivel** - está generalmente orientada al proceso de interpretación de los entes obtenidos en los niveles inferiores y se utilizan modelos y/o conocimiento a priori del dominio.

Aunque estas etapas son aparentemente secuenciales, esto no es necesario, y se consideran interacciones entre los diferentes niveles incluyendo retroalimentación de los niveles altos a los inferiores [Sucar11].

3.2. Técnicas de segmentación

En las técnicas de procesamiento de las imágenes cada píxel era transformado de un valor a otro. Estos cambios, mayormente, se realizan para facilitar la partición de la imagen en áreas de píxeles con significado. En esta nueva fase se trata de agrupar los píxeles, por algún criterio de homogeneidad, para particionar la escena en regiones de interés. Estas áreas deben de tener algún significado físico. Por tanto, la segmentación de una imagen es un proceso de extracción de los objetos de interés insertados en la escena capturada. La agrupación de los píxeles se hace a razón de que sus vecinos sean similares en criterios como de luminancia, color, texturas, movimientos. Una vez que la imagen ha sido particionada, la unidad dejara de ser el píxel para ser la agrupación de píxeles que constituye el objeto. La imagen estará definida por un conjunto de objetos, habiendo pasado de un nivel bajo a otro más elaborado o nivel medio visual. La información estará preparada para el reconocimiento e interpretación de la imagen, en la Figura 3.1 se muestra un esquema general de visión computacional donde se aprecia el lugar donde se sitúa la segmentación.

Para la segmentación de las imágenes se usan tres conceptos básicos [Dep11]:

- Similitud: los píxeles agrupados del objeto deben ser similares respecto algún criterio (nivel de gris, color, textura).
- Conectividad: los objetos corresponden a áreas de píxeles con conectividad. Las particiones corresponden con regiones continuas de píxeles.
- Discontinuidad: los objetos tienen formas geométricas que definen unos contornos. Estos bordes delimitan unos objetos de otros.

En la práctica, la imposición de estas condiciones sobre la estrategia de segmentación resulta casi imposible. Así, por ejemplo, los criterios de similitud fallan debido a la aparición

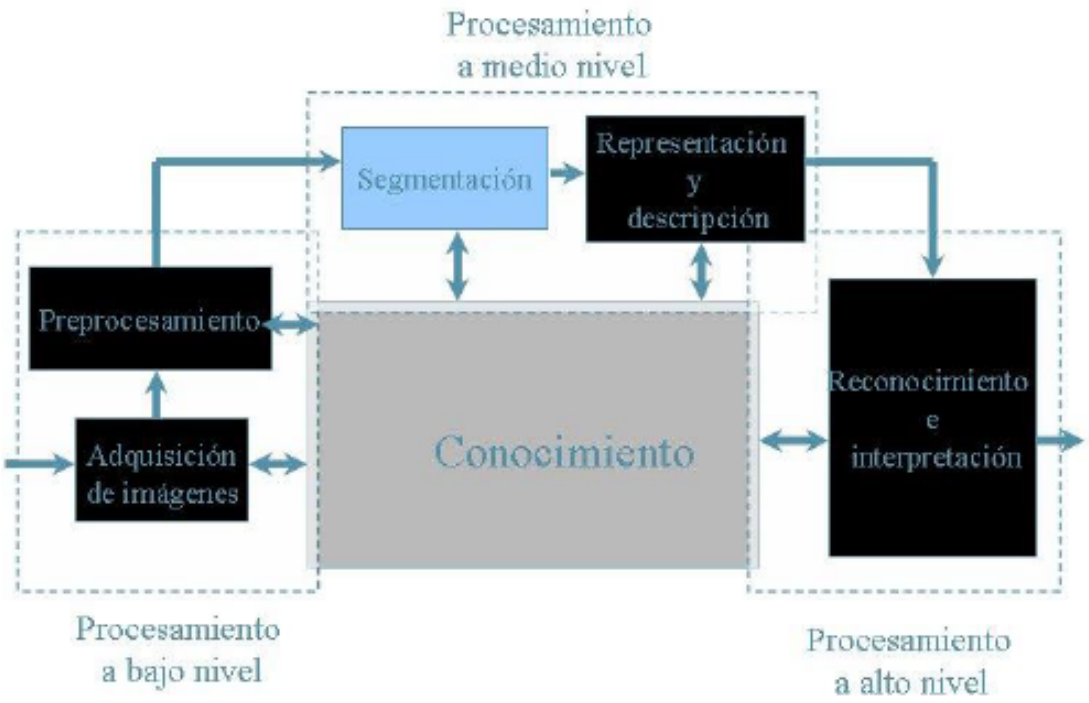

Figura 3.1: Esquema general de visión computacional, en donde se aprecia la ubicación de la segmentación en el sistema [Dep11].

del ruido, a la falta de iluminación uniforme sobre el escenario o a la creación de sombras de unos objetos sobre otros. Todas estas causas producen que algo que parecía sencillo de definir, como es alguna regla sobre la similitud, resulte impracticable de acotar. Todas ellas fracasan sobre escenas más o menos complejas. En cuanto a la conectividad, la ocultación parcial de un objeto capturado hace fallarla. Suele ser normal en el proceso de segmentación que se produzcan varias regiones de píxeles agrupados que provienen del mismo objeto físico. No sólo por las oclusiones de éste, sino también debido a los cambios de la textura del objeto, a la variación de la iluminación de la escena o al reflejo de otros objetos adyacentes. La detección de los contornos físicos suele estar plagada de errores y de discontinuidades en los bordes. Resulta extraordinariamente difícil obtener los contornos cerrados, inmunes al ruido y sin desplazamiento entre el contorno real y el obtenido, en la Figura 3.2 se muestra un ejemplo de las técnicas de segmentación sobre escenarios naturales [Dep11].

En la actual Visión computacional todavía existe mucho recorrido para las apli-

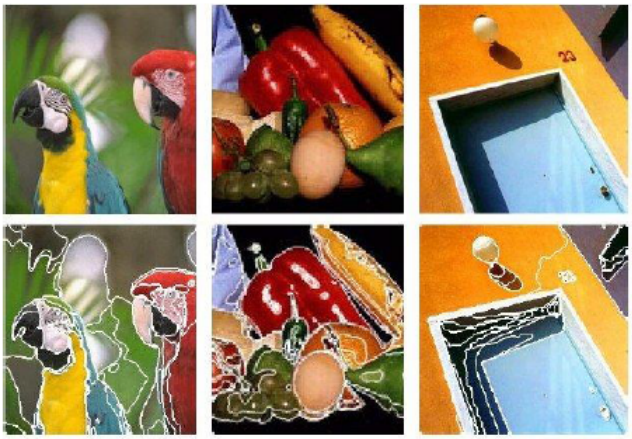

Figura 3.2: Segmentación sobre escenarios naturales [Dep11].

caciones. Las citadas dificultadas de la fase de segmentación son resueltas con una elección esmerada en la formación de la imagen, eligiendo una iluminación adecuada y simplificando la escena. Las técnicas de Visión Artificial resultan factibles para el análisis de imágenes simples y repetitivas tales como se muestran el la Figura 3.3.

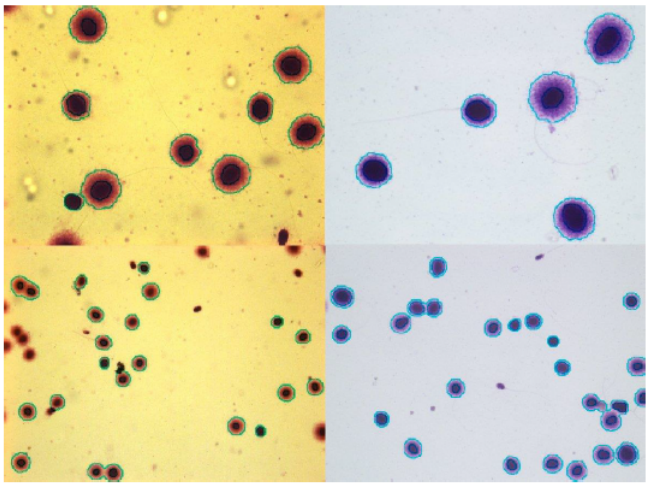

Figura 3.3: Técnicas de segmentación sobre escenarios repetitivos. Aplicaciones de segmentación sobre imágenes procedentes de microscopía [Dep11].

Existen varios métodos para la segmentación de imágenes descritos cada uno en

[Dep11] y [Sucar11] entre los cuales tenemos los siguientes:

- Detección de orillas.
- Técnicas basadas en los bordes.
- Transformada de Hough.
- Detección de líneas rectas.
- Detección de círculos.
- Transformadas de Hough generalizadas.
- Umbralización.
- Segmentación por histograma.
- Selección óptima del umbral.
- Segmentación orientada a las regiones.
- Crecimiento de regiones.
- Pirámides y árboles cuaternarios.
- División y fusión de regiones.
- Modelos espectrales.

3.2.1. Detección de orillas

Detectar orillas es una tarea particularmente importante en visión por computadora. Los límites o bordes físicos, discretizados como variaciones de intensidad, son un punto de partida para tareas de bajo nivel como detección de esquinas, bordes y compresión de imágenes; y son la base de tareas de nivel intermedio como la separación o segmentación de los diferentes objetos en una imagen.

La manera más común para detectar orillas es utilizar algún tipo de derivada, aplicado normalmente en un vecindario "pequeño". La derivada nos permite calcular las

variaciones entre un punto y su vecindario. Viendo la imagen como una función, un contorno implica una discontinuidad en dicha función, es decir donde la función tiene un valor de gradiente o derivada "alta" (ver la Figura 3.4).

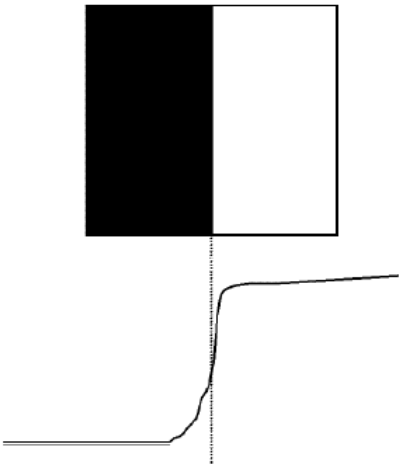

Figura 3.4: Ejemplo de discontinuidades. Se muestra una imagen con una discontinuidad en intensidad entre la parte izquierda y derecha. En la figura de abajo se gráfica la intensidad de un "corte" horizontal de la imagen (un renglón) en el que se observa el alto gradiente en la parte correspondiente a la discontinuidad.

Al apreciar detenidamente un borde en una imagen vemos que éste se encuentra integrado de "orillas locales" u orillas individuales. En visión por computadora cada una de estas orillas locales (Figura 3.5) son integradas o unidas, en etapas posteriores, en algo más útil que píxeles aislados, a estos les llamaremos bordes.

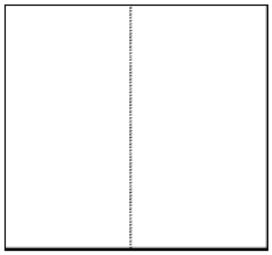

Figura 3.5: Se puede ver el borde o discontinuidad de la imagen de la Figura 3.4 como constituido por una serie de "puntos" que corresponden a orillas locales.

La detección de orillas, es bastante sensible al ruido lo cual dificulta el proceso de integración de bordes. Debido a esta dificultad han surgido una gran cantidad de técnicas

de detección de orillas y es, quizá, el tema con mayor número de artículos publicados en la literatura especializada en visión. El principal problema a lo que se enfrentan cada uno de estos trabajos es el como reconocer las orillas "visualmente relevantes", que pertenecen a contornos de interés, para diferenciarlas de otras orillas "falsas" generadas por fenómenos como ruido, sombreado, textura, etc.

Después de obtener las orillas, es común que se seleccionen de las orillas "relevantes", utilizando cierta información del contexto o del dominio. Tales técnicas "forzan" a detectar círculos, líneas, objetos largos, cambios "suaves", etc. Este post-procesamiento se conoce como task-driven o dependiente de la tarea a realizar [Sucar11].

Las técnicas de detección de orillas se pueden clasificar en [Sucar11]:

- Operadores de gradiente.
- Múltiples respuestas a diferentes orientaciones.

3.2.2. Técnicas basadas en los bordes

Detección de bordes de Canny.

Detección de bordes de Canny es un algoritmo de detección de bordes popular. Fue desarrollado por John F. Canny en 1986. Se trata de un algoritmo de varias etapas siendo descritas en [Ope15]:

- Reducción de ruido.
- Encontrar Intensidad de degradado de la Imagen.
- Supresión de mínimos.
- Umbral de interés.

Para la localización de las fronteras de los objetos en la escena se emplean las técnicas de detección de los bordes ya mencionadas. Sin embargo, esta etapa no es definitiva

para poder segmentar los objetos presentes en la imagen. La presencia de ruido, el efecto de las sombras, la falta de iluminación uniforme y un largo etcétera de causas, produce que los contornos no sean del todo continuos y cerrados sobre los objetos. Se requiere otra etapa de post-procesamiento. Esta nueva fase emplea los resultados de la detección de bordes para elaborar las fronteras de los objetos. Se trata de agrupar los píxeles etiquetados como bordes, de la etapa anterior (de detección de bordes), empleando la propiedad de conectividad. Para que un píxel etiquetado como borde se defina como píxel frontera de un objeto se necesitará que otros píxeles bordes tengan similar dirección y módulo del gradiente. Dos píxeles serán considerados pertenecientes a una misma frontera si presentan alguna condición de conectividad y las diferencias entre sus gradientes no superan un determinado umbral [Dep11].

Evidentemente, en la construcción de la frontera aparecerá ruido, habrá píxeles que constituyen los bordes de los objetos y otros que son defectuosos y que han aparecido por una alta variación local de intensidad debido a sombras, cambios de iluminación. Hay varios planteamientos para su implementación. Algunos autores emplean técnicas de crecimiento de regiones mencionadas en [Dep11], pero que básicamente se trata de partir de un píxel borde semilla ir agrupando píxeles con conectividad siempre y cuando cumplan algún criterio de homogeneidad, por ejemplo, el expresado en la ecuación 3.1.

$$\begin{aligned} ||G_1| - |G_2|| &< T_M \\ |\theta_1 - \theta_2| &< T_A \end{aligned} \tag{3.1}$$

Donde G_i y θ_i se ha denotado el módulo y argumento del gradiente del píxel i, siendo $i = 1$ ó 2, indicando dos píxeles vecinos.

Otros autores emplean un marco de trabajo basado en la optimización. Realizan un grafo de los posibles caminos que pueda llevar la frontera del objeto y mediante una función de coste, eligen la mejor solución. Un planteamiento alternativo es la agrupación de los píxeles que cumplen un cierto criterio en una primitiva más elaborada, como puede ser un pequeño segmento con orientación. Posteriormente, se procede a encadenar estas primitivas dando paso a la delimitación de los objetos[Dep11].

Un campo muy atractivo en esta materia son los contornos activos. Se trata de emplear una curva cerrada que vaya adaptándose dinámicamente hasta alcanzar la frontera del objeto como se muestra en la Figura 3.6 [Dep11].

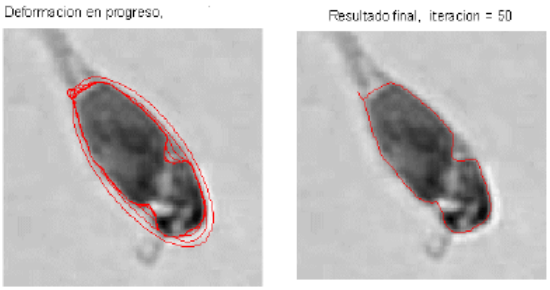

Figura 3.6: Ejemplo de contornos activos.

3.2.3. Transformada de Hough

Este algoritmo trata de detectar formas geométricas sencillas en una imagen. En su procesamiento emplea el conjunto total de la imagen, haciéndolo robusto ante la presencia del ruido o discontinuidades de las etapas previas. De hecho, para su ejecución requiere de una imagen binarizada en la que se ha seleccionado previamente los bordes. Por tanto, la entrada al algoritmo de Hough es una imagen en la que se le ha aplicado un detector de bordes. Hough, desde una perspectiva global, intentará extraer las primitivas de más alto nivel como pueden ser líneas, elipses o cualquier tipo de curva parametrizada o no. El mayor inconveniente es su alto coste computacional [Dep11]. En la figura 3.7 Se muestra un ejemplo para detectar una circunferencia, donde se aprecia una imagen antes de aplicarle la transformada de Hough y otra imagen donde ya se le aplica la transformada de Hough.

3.2.4. Modelos Espectrales

La transformada de Fourier es adecuada en describir información “global” en la imagen, en especial patrones periódicos. Este es el caso de las texturas, generalmente, por lo que los modelos espectrales proveen buenas características para su descripción y clasificación. En particular, hay 3 características del espectro que son adecuadas para la descripción

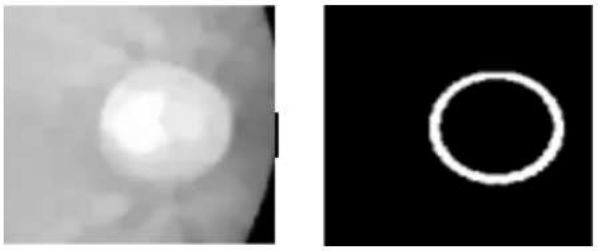

Figura 3.7: Detección de circunferencia. Imagen a) Antes de aplicar transformada de Hough. Imagen b) Después de aplicar trasformada de Hough (circunferencia de Hough) [DelaFuente12].

de texturas [Sucar11]:

- La amplitud de los picos prominentes dan la dirección principal de los patrones en la textura.
- La localización de los picos en frecuencia indican el periodo espacial de los patrones.
- Eliminando componentes periódicas mediante filtros en Fourier, se pueden dejar sólo las componentes a-periódicas a las que se les aplican técnicas estadísticas.

Estas características son más fáciles de detectar convirtiendo el espectro a coordenadas polares, en la Figura 3.8 se muestran algunos ejemplos [Sucar11].

La Figura 3.9 ilustra un ejemplo de segmentación de texturas en base a características obtenidas con filtros gaussianos a diferentes escalas (multi-escala). A estos atributos de escala (parte intermedia de la figura) se les aplicó un proceso de regularización para realizar la segmentación de texturas.

3.3. Seguimiento de objetos

Un uso común del análisis de una secuencia de imágenes en movimiento es el seguimiento (tracking) de objetos en las imágenes. Esto tiene múltiples aplicaciones prácticas, como el seguimiento de personas o vehículos para fines de seguridad, el seguimiento de las partes del cuerpo de una persona para reconocer actividades o ademanes, y el seguimiento de vehículos terrestres, marítimos o aéreos en aplicaciones militares [Sucar11].

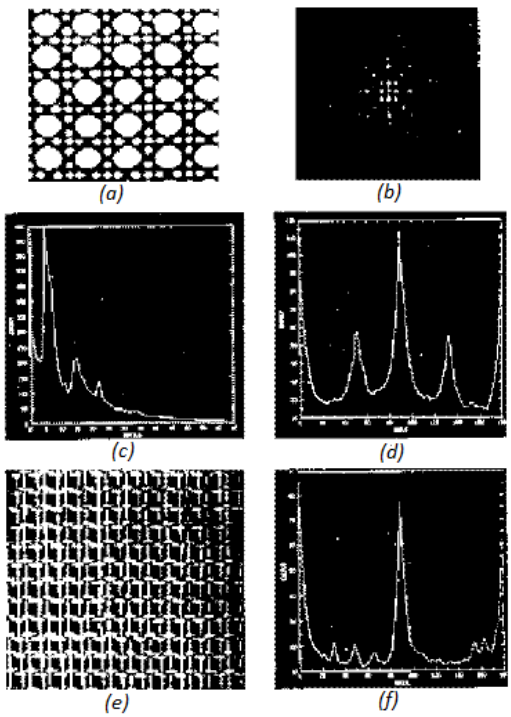

Figura 3.8: Ejemplos de espectros en coordenadas polares para diferentes texturas periódicas: (a) imagen de una textura, (b) espectro, (c) gráfica del espectro en r (radio), (d) gráfica del espectro en teta (ángulo), (e) imagen de otra textura, (f) gráfica del espectro en teta (ángulo).

3.3.1. Seguimiento

El seguimiento en una secuencia de imágenes consiste en determinar la posición y velocidad de un punto (o de una región u objeto) en una imagen, dada su posición y velocidad en una secuencia anterior de (una o más) imágenes. El seguimiento se puede realizar en base a diferentes atributos de la imagen, en particular se pueden distinguir las siguientes clases de objetos [Sucar11]:

- Modelos rígidos bidimensionales o tridimensionales de objetos.
- Modelos deformables.
- Regiones.

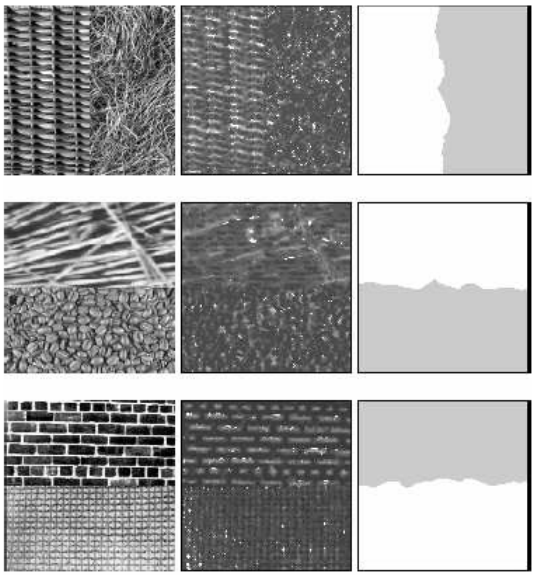

Figura 3.9: Se ilustran 3 ejemplos de mosaicos con dos texturas cada uno. Del lado izquierdo se tiene la imagen original, en la parte media los atributos de escala de las texturas, y en la parte derecha la separación de las texturas con un nivel de gris diferente para cada clase [Sucar11].

- Características de la imagen (puntos, líneas, esquinas, etc.).

Un ejemplo es el seguimiento de puntos: Es cuando el objeto es representado por puntos durante frames consecutivos y la asociación de los puntos está basada en el estado del objeto previo, incluyendo en ocasiones posición y movimiento. Este tipo de seguimiento contiene métodos como el Filtro de Kalman y Filtro de partículas [Cuevas13].

Existen técnicas robustas para el seguimiento de objetos, tales como flujo óptico o la técnica del vector de corrimiento de medias. Estas técnicas son las más utilizadas para el seguimiento de objetos en movimiento. La ventaja que ofrecen este tipo de técnicas es la detección simultánea con el seguimiento, con lo cual se obtiene un ahorro en el tiempo de procesamiento. La gran desventaja, la cual es de vital importancia, es que son muy susceptibles al ruido, a condiciones de oclusión y a fallan en un background complejo. Esto se puede solucionar mediante una serie de heurísticas y reforzando el mecanismo con alguna otra técnica de detección, haciendo el algoritmo más robusto.

También existen técnicas para sistemas de control, como lo son los filtros adaptativos, entre los cuales destacan los fíltros de Wiener, de Kalman y los filtros $g-h-k$ Bookner [Brookner98]. De acuerdo con el tipo de sistema, es el filtro a utilizar. En el caso de un comportamiento determinístico se utiliza el filtro de Wiener y los filtros $g-h-k$. En el caso de sistemas dinámicos aleatorios, es común hacer uso del filtro de Kalman. Para el caso de seguimiento de objetos lo más común es utilizar el filtro de Kalman, pues el comportamiento del objeto es más bien aleatorio. En la Figura 3.11 se puede observar una taxonomía de las diversas técnicas existentes para el seguimiento de objetos [Cuevas13].

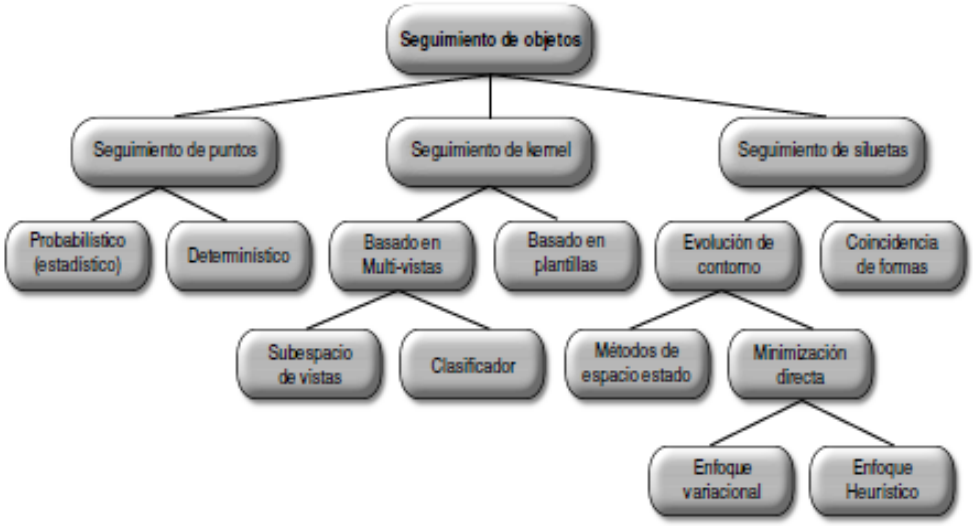

Figura 3.10: Taxonomía de los métodos de seguimiento de objetos. Yilmaz et al [Yilmaz06].

3.4. Algoritmo para seguimiento en video

El algoritmo de Meanshift es utilizado para encontrar y rastrear objetos en los videos, por mencionar alguna de sus aplicaciones. Este procedimiento fue originalmente propuesto por [Fukunaga75] en el año de 1975.

3.4.1. Algoritmo Meanshift

El algoritmo meanshift puede ser utilizado para el seguimiento visual de un objeto en una secuencia de imágenes. La intuición detrás de meanshift es simple, dado un conjunto

de puntos, (Puede ser una distribución de píxeles como retroproyección de histograma), se asigna una pequeña ventana (no necesariamente rectángular, puede ser un círculo o cualquier otra forma), la cual se debe mover a la zona de densidad de píxeles máximo (o número máximo de puntos). La Figura 3.11 ilustra un ejemplo del algoritmo de meanshift [Ope15].

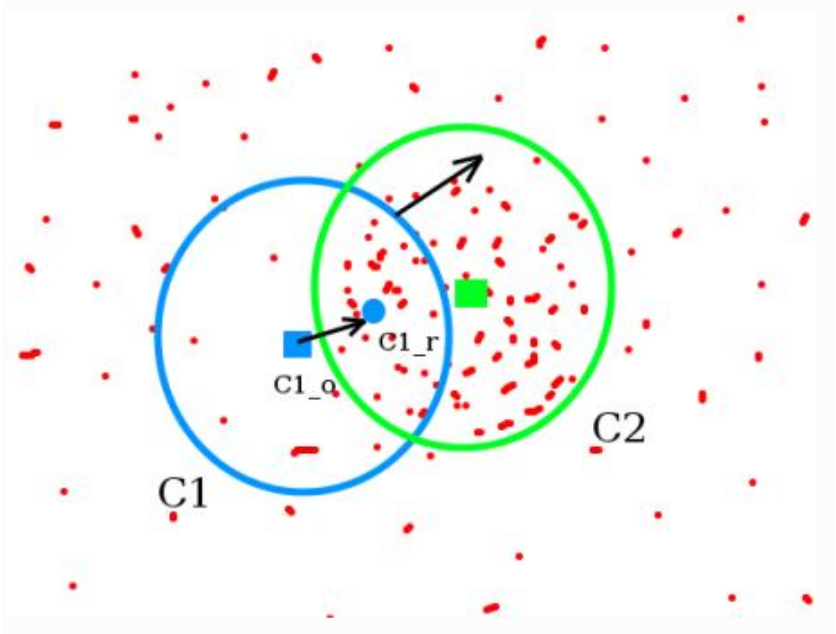

Figura 3.11: El círculo inicia en una posición cualquiera y se mueve a otra posición donde el conjunto de puntos es mayor, o la densidad es máxima [Ope15].

De la Figura 3.11 en la venta inicial se muestra un círculo de color azul con el nombre "C1". Su centro o centroide original está marcado con el rectángulo de color azul, llamado "C1_o". Pero si se encuentra el centro de gravedad de los puntos dentro de esa ventana, se obtiene el punto "C_r" (marcado como un pequeño círculo de color azul), que es el centro de gravedad real de la ventana. Se observa que ambos centros de gravedad no coindicen. Así que se debe mover la ventana de tal manera que el circulo de la nueva ventana coincida con el centroide anterior. Se procede a encontrar una vez más el centro de gravedad, y mover la ventana nuevamente al centro de gravedad que representa la mayor densidad de píxeles, se continua con las iteraciones de tal forma que se busca que el centro de la ventana y su centroide se encuentren en la misma ubicación (o con un pequeño error

deseado). Así que, finalmente, lo que se obtiene es una ventana con máxima distribución de píxeles, en la figura 3.11 esta marcado con el círculo verde, llamado "C2" y con el centroide marcado por el rectángulo del mismo color [Ope15].

Capítulo 4

Diseño e implementación del sistema

4.1. Materiales y equipo

Los dispositivos que conforman este sistema de visión computacional para el conteo de personas son:

- Un equipo de cómputo que sea capaz de manipular secuencias de video.
- Una cámara web con resolución mínima de 640 x 480 píxeles, como las que se muestran en la Figura 4.1 Nota: Para poder utilizar una mayor resolución se requiere mayor poder de cómputo.
- Librerías de código abierto OpenCV y lenguaje de programación Python, incluidas las librerías Numpy para el mejor manejo de listas y matrices.

4.1.1. Equipo de cómputo

Un equipo de cómputo capaz de manipular secuencias de video, cuenta con las siguientes especificaciones:

- Procesador: Intel(R) Pentium(R) Dual-Core CPU T320 a 2.00GHz

Figura 4.1: Cámara web para grabar video [tea15].

- Memoria RAM: 2.00 GB
- Disco duro: 232.89 GB

4.1.2. Cámara web

El elemento mínimo de una pantalla es el píxel, el cual no tiene un tamaño exacto ni definido, sino que se adapta al tamaño de la pantalla en que se muestra: por ejemplo, "una pantalla de 29' es 4 veces el tamaño de una de 14' ", sin embargo, siguen teniendo la misma cantidad de píxeles (720 de ancho por 480 de alto) lo que hace que lógicamente los píxeles de la pantalla de 29' "son mucho más grandes que en la pantalla de 14" (esto explica por qué siempre se ve mejor en pantallas más pequeñas). Si comparamos la misma imagen en diferentes tamaños, el resultado sería el que muestra la Figura 4.2 [gui10].

El nombre corto para una de las categorías de los modos de vídeo se específica cómo 480i, 720p, y 1024i. El número 480, 720, o 1024 representa el número de líneas verticales de resolución de pantalla, mientras que la letra **p** significa barrido progresivo [Wikipedia15] y la letra i significa barrido entrelazado, por el momento no es necesario saber más sobre que

Figura 4.2: Misma imagen en diferentes tamaños, mostrando la nomenclatura de los modos de video.

es progresivo o entrelazado, más adelante se explicará.

Entonces también quiere decir que una imagen de 1920 x 1080 píxeles es casi 4 veces el tamaño que la de 720 x 480 píxeles. ¿Qué sucede si tomamos las otras imágenes de la Figura 4.2 (480 y 720 píxeles) y las llevamos a una resolución de 1080 píxeles? El resultado lo podemos ver en la Figura 4.3 [gui10].

Como se puede apreciar, la resolución y calidad de imagen disminuyó considerablemente debido a que en el mismo espacio en que una imagen de 1920 x 1080 píxeles es mostrada, se llevó una de 720 x 480 píxeles y otra de 1280 x 720 píxeles, haciendo que los píxeles crezcan 4 veces su tamaño en el primer caso y 2 veces en el segundo. Este es el mismo efecto que se aprecia al ver señales de definición estándar (SD TV) en un televisor de alta definición, ya que la señal de origen viene en 720 x 480 píxeles (o en algunos casos 640 x 480 píxeles ó 520 x 480 píxeles en el caso de SD digital) y se intenta mostrar en una pantalla con muchos más píxeles [gui10].

Se eligió trabajar con una resolución 640 x 480 píxeles, por ser del tamaño suficiente para poder analizar el video que se esta tomando y poder procesarlo, con una resolución inferior podríamos perder información relevante para la detección de los objetos (Contornos o rostros), y con una mayor resolución podríamos tener mucha más información a la hora de analizar el video pero para ello se requiere una mayor capacidad de procesamiento o poder de cómputo.

Figura 4.3: Imágenes de diferentes resoluciones llevadas a una resolución de 1080 píxeles.

4.1.3. Librería OpenCV

En la parte del software en el sistema, se utilizan las bibliotecas de código abierto que constituyen a OpenCV. OpenCV fue elaborado para proporcionar una infraestructura común para aplicaciones y el procesamiento de imágenes de visión computacional [Laganière11].

Se elige trabajar con las funciones de OpenCV por su fácil y rápida manipulación e implementación de sistemas de visión por computadora, las cuales nos ahorraran tiempo de desarrollo y eficiencia en el diseño y elaboración del código a programar para el sistema de visión computacional planteado en este trabajo.

4.1.4. Lenguaje de programación Python y biblioteca NumPy

El desarrollo del código del programa está elaborado en el lenguaje de programación Python, y la biblioteca NumPy. Python es un lenguaje de programación de propósito general, orientado a objetos el cual cuenta con las siguientes características [Duque11]:

- Simplicidad.
- Versatilidad.
- Rapidez de desarrollo.
- Independiente de la plataforma.

Python es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad en el procesamiento [Duque11].

NumPy es una extensión de Python, que le agrega mayor soporte a la manipulación de vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices [Num15].

4.2. Diseño del sistema

4.2.1. Descripción general del sistema de visión computacional para el conteo de personas

En el sistema contador de personas mediante visión computacional la cámara es la encargada de capturar el video siendo este el primer paso a realizar, después la secuencia de video es enviada al equipo de cómputo (CPU), donde es procesada y analizada por el Software desarrollado para el conteo de personas, y finalmente los resultados obtenidos son recibidos y mostrados por una pantalla o monitor, además se pueden almacenar en un sistema de bases de datos, en la Figura 4.4 se muestra un diagrama que indica la forma en cómo se relacionan los dispositivos físicos que componen el sistema de visión.

Figura 4.4: Diagrama esquemático que relaciona los componentes físicos del sistema de visión.

4.2.2. Colocación de la cámara

En este método para el desarrollo del sistema de visión computacional para el conteo de personas, no se considera la profundidad de las imágenes, por lo que la cámara se coloca en el techo y es enfocada al suelo para analizar el desplazamiento de las personas en el área visible, un ejemplo de como se coloca la cámara para la captura del video en una posición adecuada se puede ver en la Figura 4.5.

4.3. Implementación del software

Se realizaron varias implementaciones de software para realizar la tarea planteada. Dentro de la parte de seguimiento de las personas identificadas, se proponen dos métodos nuevos que utilizan la identificación de contornos como base para el seguimiento, y una tercera implementación en donde se utilizó el algoritmo Meanshift [Ope15, Fukunaga75]. Las implementaciones del sistema se definen de acuerdo al método de seguimiento utilizado de la siguiente forma:

- Contorno de mayor área
- Envolvente de los contornos significativos

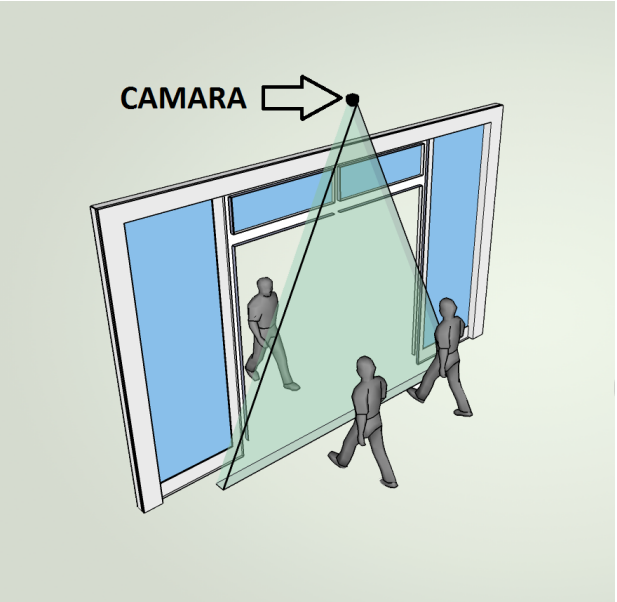

Figura 4.5: Camara colocada en la parte superior del acceso capturando video apuntando hacia el suelo [blu14].

- Algoritmo Meanshift

4.3.1. Implementación del sistema con el método de seguimiento por el contorno de mayor área

4.4. Filtro Adaptativo

Diagrama de flujo

La Figura 4.6 muestra el diagrama de flujo del sistema de visión computacional para el conteo de personas desarrollado por el método que identifica el contorno de mayor área para realizar el seguimiento. Las partes principales son las siguientes:

- Obtención del fondo de la imagen (background).
- Filtrado de ruido en la imagen y búsqueda de contornos candidatos.
- Determinar las coordenadas (x, y) de los puntos del contorno de mayor área encontrado.

- Seguimiento de contornos por posición (x, y) de referencia.
- Conteo de trayectorias que pasan la referencia definida.

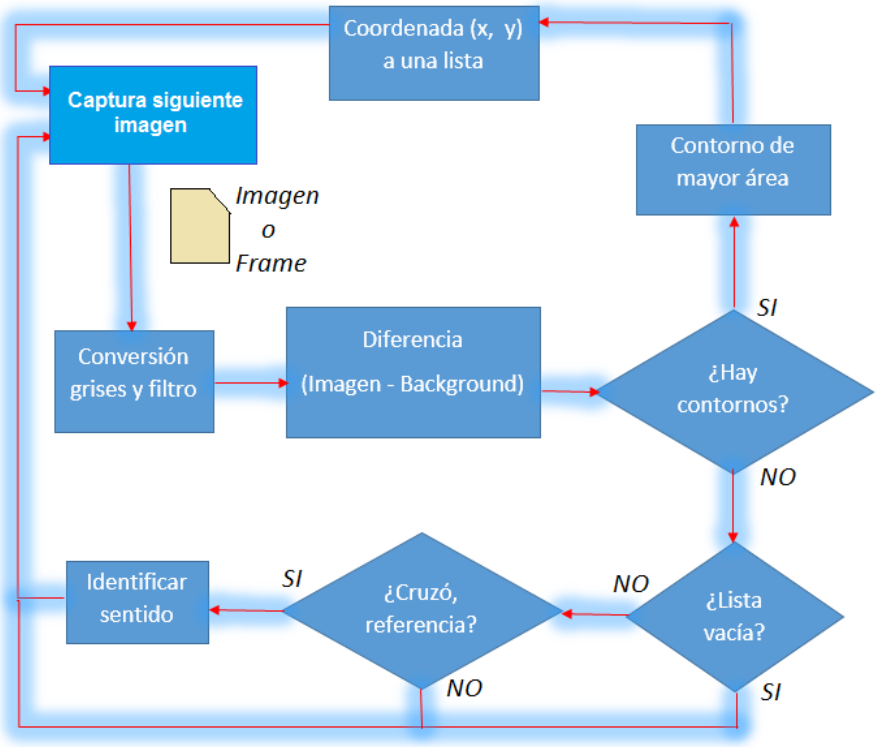

Figura 4.6: Diagrama de flujo del sistema de visión computacional para el conteo de personas por el método de detección de contornos de mayor área.

Obtención de la imagen de fondo o Backgroung

Al inicio del programa se realiza una primera captura u obtención del primer frame con la cámara, un ejemplo es mostrado en la Figura 4.7, el procesamiento realizado sobre esta imagen es:

a) se convierte a escala de grises, debido a que el procesamiento es más eficiente en este espacio de color y así poder ahorrar poder de cómputo, segmentando la imagen de una forma más rápida. La imagen obtenida en escala de grises se muestra en la Figura 4.8.

Figura 4.7: Primera imagen (frame) original capturada por la camara.

Figura 4.8: Primera imagen (frame) convertida a escala de grises.

b) Una vez obtenida la imagen en escala de grises se le aplica un filtro de umbralización adaptativo, que permite en forma automática sobrellevar la problemática de iluminación, el resultado obtenido se denomina imagen de fondo o background. La Figura 4.9 muestra el resultado obtenido al aplicar el filtro a la imagen mostrada en la figura 4.8 y posteriormente aplicarle el filtro de umbralizacion adaptativo.

Captura de video o frames posteriores al primero

Una vez obtenida la imagen de fondo o Background, cada frame subsecuente es procesado en forma similar al frame o imagen de fondo (Background), el propósito es obtener la diferencia entre ellos, determinando que si existe una gran diferencia, está se define como

Figura 4.9: Imagen de fondo o Background. Primera imagen (frame) convertida a escala de grises y filtro de umbralizacion adaptativo.

un movimiento en la imagen, mientras que una diferencia mínima nos indica que no existe movimiento. Así este mecanismo simple nos permite la identificación de movimiento en el sistema y además nos proporciona únicamente las partes de la imagen en donde realizar la búsqueda de contornos, específicamente se refiere a la diferencia de las imágenes. La Figura 4.10 muestra un ejemplo de detección de movimiento. La imagen original de un frame subsecuente donde se detectó movimiento (pasando una persona): Figura 4.10 a) la imagen de background, Figura 4.10 b) y la imagen de donde se obtuvo la diferencia, Figura 4.10 c) indicando una región en donde se detecta movimiento.

Análisis de contornos

Los contornos son elementos en una imagen que nos permiten; Detectar las bordes de los objetos en este caso los de las personas, de esta manera se logran obtener las regiones de interés de cada frame, con ellos logramos hacer la segmentación de la imagen para que los contornos pueden ser analizados con mayor eficiencia, trabajando sobre una sola región de la imagen y saber así si lo que pasa frente a la camara es una persona o no. Al utilizar este método nos es más eficiente y rápido a la hora de detectar a las personas debido a que al analizar solo regiones de interés, se ahorra poder de cómputo debido a que es menor el procesamiento que tiene que realizar el procesador de la computadora.

Los contornos identificados son analizados de la siguiente manera:

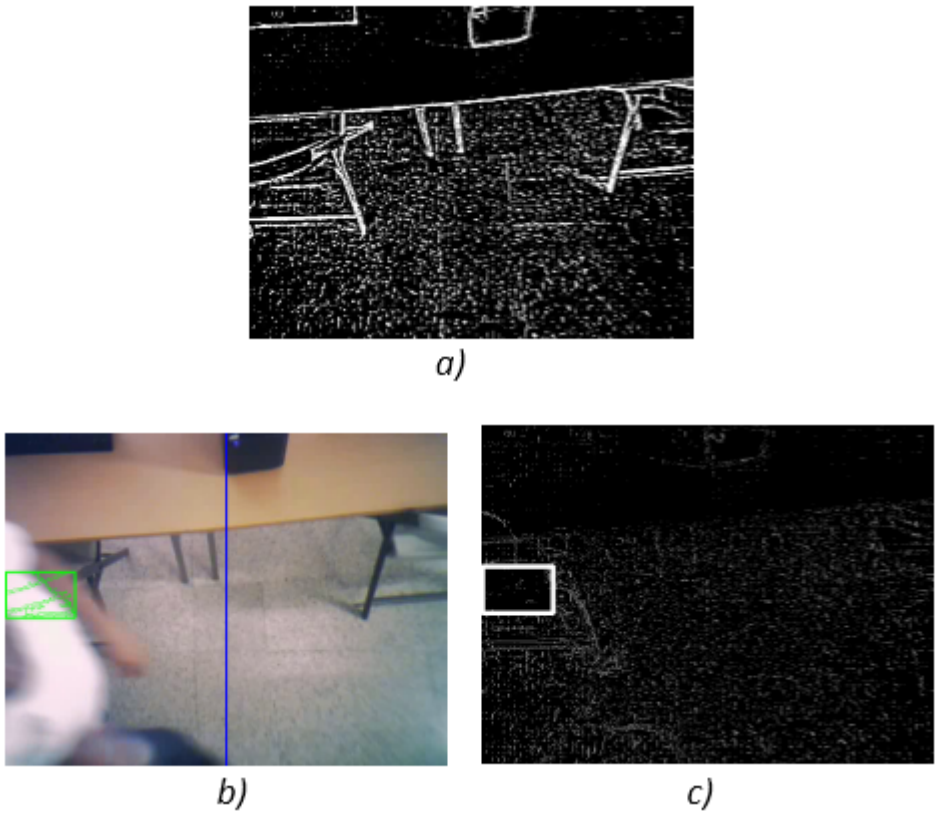

Figura 4.10: Proceso para identificar el movimiento. En a) se muestra el fondo de la imagen o Background. b) Frame subsecuente donde se detecto movimiento, señalado con un recuadro de color verde una de las áreas en movimiento. c) Diferencia obtenida del Frame subsecuente ("Imagen b) convertida a escala de grises y aplicándole el filtro de umbralización adaptativo y la imagen de Background"), se señala una parte de la diferencia con un recuadro color blanco, que al igual que en b) se refiere una parte del área en movimiento detectada.

1. Se buscan contornos en la imagen (diferencia, obtenida del frame subsecuente menos el de fondo o Background), y se obtiene el área de cada uno de ellos, identificando la posición (x, y) en la imagen de aquel con mayor área. Para eliminar contornos que no son candidatos, se estableció un valor de umbral para el área mayor a 800, valor establecido mediante la experimentación. En la Figura 4.11 se puede ver el área pintada de color verde por un recuadro del máximo contorno encontrado en el frame, mostrándose en la imagen que observa el movimiento (diferencia), y el frame original en el cual se aprecia más claramente a la persona desplazándose en la escena.

2. Por cada contorno de mayor área determinado en los frames, se almacena un historial

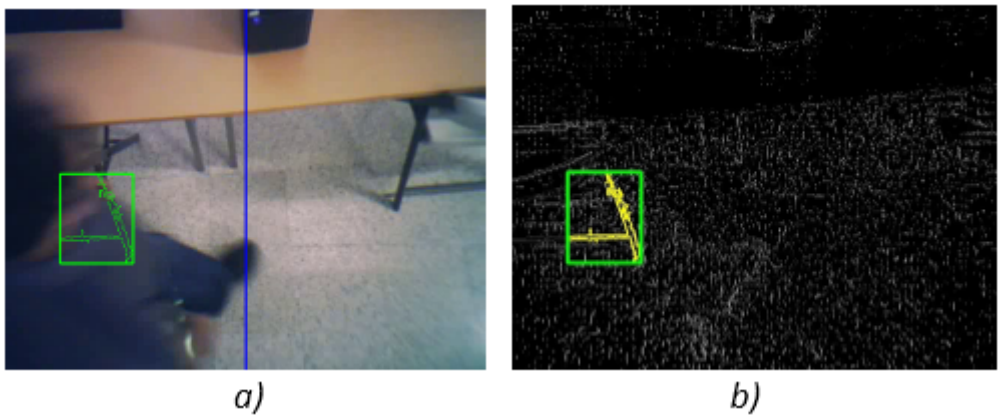

Figura 4.11: Identificación de contornos significativos. a) frame original con su máximo contorno encontrado (pintado color verde) y encerrado por un recuadro del mismo color. En b) se observa la imagen diferencia, señalando su máximo contorno encontrado (pintado color amarillo) y encerrado por un recuadro de color verde.

de posiciones (x, y), encontradas durante el tiempo donde se observa desplazamiento e identificación de contornos candidatos (mayores al umbral definido), se almacenan mediante la concatenación en una lista L. En la Figura 4.12 se muestra la secuencia de video que señala el recorrido de una persona, mostrando los contornos encontrados con área mayor, los cuales son identificados y rodeados por un recuadro de color verde, almacenando las coordenadas de la esquina superior izquierda de cada recuadro.

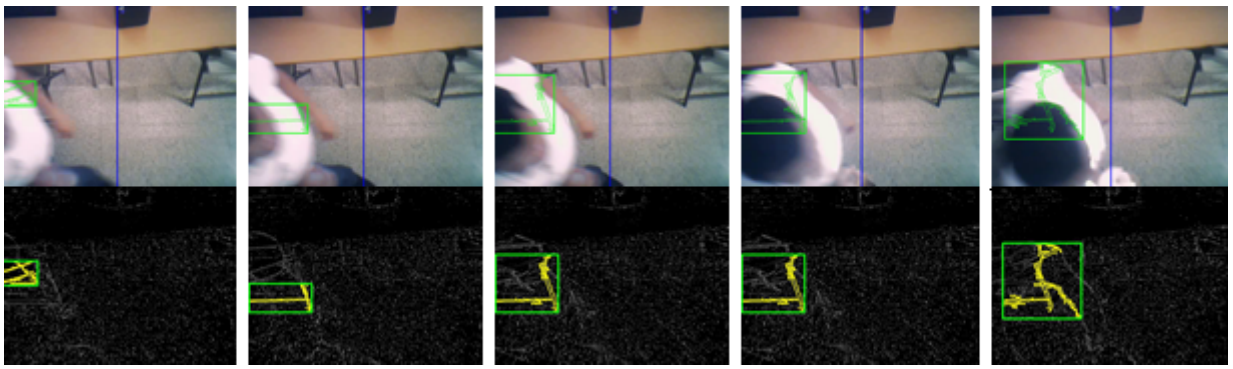

Figura 4.12: En la parte superior de la imagen se observa la secuencia de los frames originales (sin ningún filtro) que conforman una parte del video donde se observan los contornos encontrados rodeados por un recuadro de color verde. En la parte inferior de la imagen se observa la secuencia de los frames obtenidos mediante la diferencia (Frame Background - Frames subsecuentes con filtros) que conforman una parte del video donde se observan los contornos encontrados rodeados por un recuadro de color verde.

3. Finalmente, de la lista obtenida con las coordenadas (x, y) se obtiene el valor máximo y mínimo de los valores que representan la coordenada x del conjunto de puntos almacenados en la lista.

$$x_{min} = min(x) \in L$$

$$x_{max} = max(x) \in L$$

La Figura 4.13 muestra una imagen en la cual se puede apreciar los puntos de donde son obtenidas las coordenadas (x_{min}, y_{min}) y (x_{max}, y_{max}) de cada frame que contiene contornos.

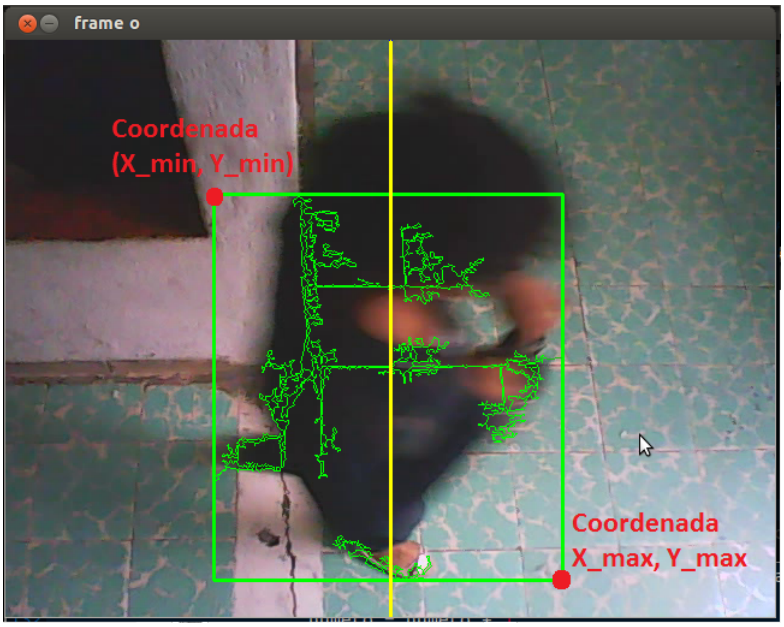

Figura 4.13: Imagen de un frame donde se ven envueltos todos los contornos encontrados por un rectángulo color verde. Se muestran los puntos pintados de color rojo de las coordenadas (x, y) máximas y mínimas.

Se obtiene la diferencia entre los valores $\{x_{min}, x_{max}\}$ para determinar si se describe una trayectoria que cruzó por la referencia definida (línea imaginaria en la mitad de la imagen), es decir, se determina que una persona paso por el acceso (frente a la cámara), ecuación 4.1.

$$d_x = x_{max} - x_{min} \tag{4.1}$$

Para determinar el sentido en que se desplazan las personas identificadas, se realiza una sumatoria con todos los elementos que corresponden a la coordenada x en los puntos de la lista, ver ecuación 4.2. Se obtiene la dirección en la que cruza la persona por el acceso mediante el resultado de la sumatoria, el cual nos proporciona un valor positivo o negativo. Si $s > 0$, es decir es positivo, indica que el desplazamiento se hizo hacia la izquierda, mientras que cuando $s < 0$, el valor de la sumatoria es negativo, el desplazamiento que se observó es hacia la derecha.

$$s = \sum_{i=1}^{n} x_i - x_{i-1} \tag{4.2}$$

En la ecuación 4.2, n se refiere al número de elementos en la lista que almacena las coordenadas de los puntos (x, y).

4.4.1. Implementación del sistema con el método de seguimiento por la envolvente de contornos.

En esta propuesta para el seguimiento de personas, que además puede aplicarse de igual forma al seguimiento de objetos, se plantea obtener el área que cubre a todos los contornos que se producen al momento de identificar el movimiento en la escena.

Diagrama de flujo

La Figura 4.14 muestra el diagrama de flujo para la implementación del sistema de visión computacional que utiliza para el seguimiento de las personas identificadas, el nuevo planteamiento que encuentra la envolvente de los contornos, las partes principales son las siguientes:

- Obtención del fondo de la imagen (background).
- Filtrado de ruido en la imagen y búsqueda de contornos candidatos.
- Determinar las coordenadas (x, y) representativa de los puntos en la envolvente de todos los contornos encontrados.

- Seguimiento de contornos por posición (x, y) de referencia.
- Conteo de trayectorias que pasan la referencia definida.

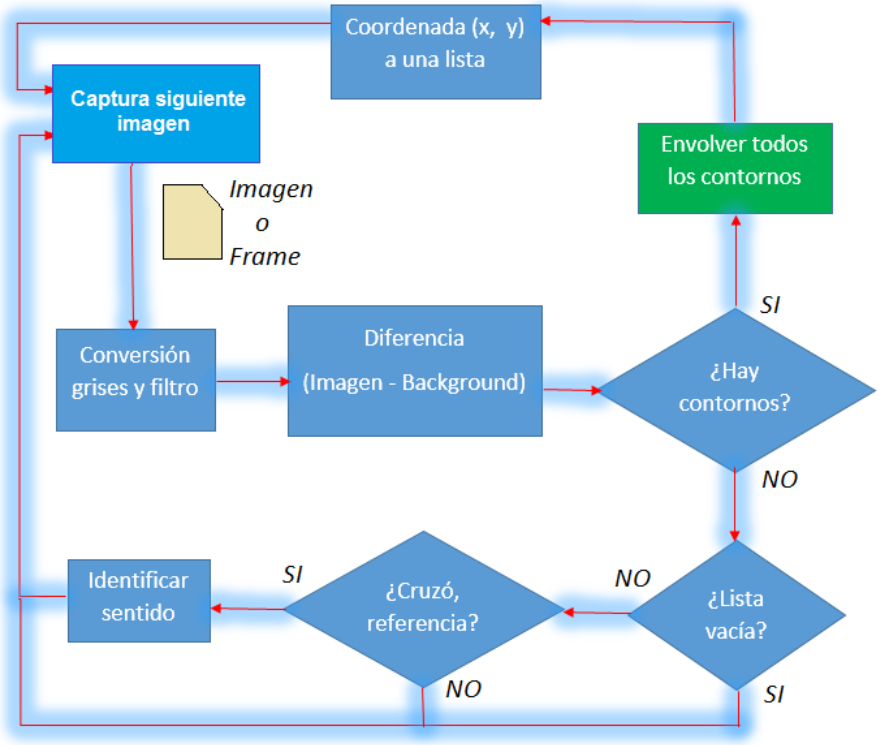

Figura 4.14: Diagrama de flujo del sistema de visión computacional para el conteo de personas por el método de detección y envolvente de contornos.

Este método (de seguimiento por la envolvente de contornos) es muy similar al método descrito en 4.3.1, la parte en la que cambia principalmente es que este método ahora no encuentra el contorno de mayor área, sino este método lo que hace es envolver todos los contornos encontrados, para realizar el seguimiento de video con todos los contornos encontrados y no solo con el de mayor área como lo hace el método de seguimiento por el contorno de mayor área.

Envolvente de los contornos

Para determinar la envolvente de los contornos, se establece a $C = \{c_1, c_2, \ldots, c_n\}$ como el conjunto de todos los contornos encontrados en una imagen diferencia I_d. Sea la

tupla $R_{c_i}(x, y, w, h)$ el rectángulo que circunscribe el contorno $c_i \in C$, en donde x, y corresponde a la coordenada superior izquierda del rectángulo en la imagen I_d. w, h corresponden al ancho y alto respectivamente del rectángulo que circunscribe a c_i. La envolvente se define por dos puntos P_1 y P_2, que representan los coordenadas de las esquinas opuestas del rectángulo que cubre todos los contornos en C, obtenidos por las ecuaciones 4.3 y 4.4.

$$P_1 = min(x, y) \in R_{c_i} \tag{4.3}$$

$$P_2 = max(x, y) \in R_{ci} \tag{4.4}$$

Se almacenan las coordenadas de P_1 en la lista L, la cual es utilizada posteriormente para identificar el cruce de la referencia establecida sobre la imagen I, de la misma forma como se hizo con el método 4.3.1.

4.4.2. Implementación del sistema por el método de seguimiento de video con el algoritmo de Meanshift

En está implementación se utilizó el algoritmo descrito en la sección 3.4.1. El punto de partida que se debe proporcionar al algoritmo $meanshift$ se establece por medio del ROI inicial que se obtiene por el método del contorno con mayor área descrito en 4.3.1.

Diagrama de flujo

La Figura 4.15 muestra el diagrama de flujo de la implementación del sistema de visión computacional para el conteo de personas utilizando el algoritmo $meanshift$, del cual las partes principales son las siguientes:

- Obtención del fondo de la imagen (background).
- Filtrado de ruido en la imagen y búsqueda de contornos.
- Detección del ROI inicial.
- Seguimiento de video por Memshift a partir del ROI inicial.

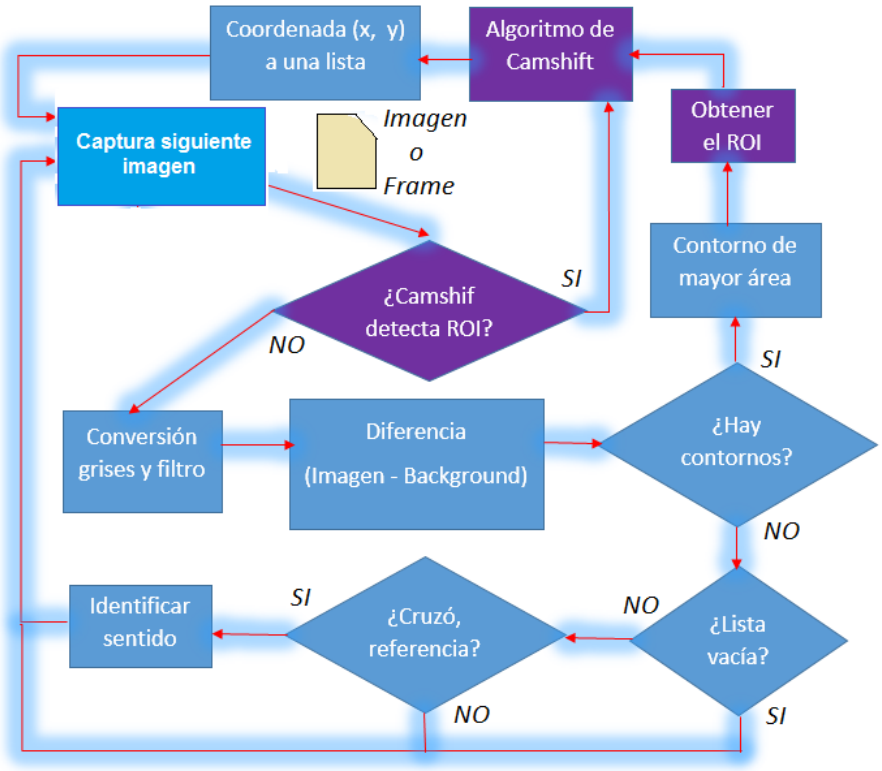

Figura 4.15: Diagrama de flujo del sistema de visión computacional para el conteo de personas utilizando el método de meanshift para el seguimiento del objeto identificado.

- Conteo de trayectorias que pasan la referencia definida.

El algoritmo $meanshift$ retorna la tupla $W(x, y, w, h)$, que representa la posición y tamaño de la ventana (rectángulo) que se sigue mientras existe movimiento en la secuencia de video. Se almacenan las posiciones (x, y) en una lista L y se procede al igual que en el método descrito en 4.3.1 para determinar el cruce de la referencia y el sentido del desplazamiento.

Capítulo 5

Experimentación y resultados

Las pruebas que se ralizaron al sistema consistieron en evaluar 3 algoritmos diferentes para el seguimiento de los objetos identificados, los cuales son:

1. Algoritmo de seguimiento por el contorno de mayor área.
2. Algoritmo de seguimiento por la envolvente de contornos.
3. Seguimiento de video con el algoritmo de Meanshift.

Se presentan resultados de la eficiencia con que se contaron las personas en cada uno de los escenarios o videos de prueba.

5.1. Descripción del conjunto de videos utilizados

Se capturaron 3 videos, cada uno de ellos en diferentes ambientes interiores, observando diferente intensidades de iluminación cuando se realizó la captura. Los nombres y una descripción física de los ambientes en donde se realizaron las capturas son los siguientes:

- Laboratorio: El laboratorio cuenta con poca luz natural, por lo cual se utilizo luz artificial, lo que permitió proporcionar una iluminación adecuada para poder realizar las pruebas con el sistema de conteo de personas, debido a que la cámara utilizada no cuenta con dispositivo alguno para capturar video con poca iluminación.

- Casa: La casa cuenta con suficiente luz natural, por lo cual no se utilizó luz artificial, está cantidad de luz fue suficiente para llevar a cabo las pruebas planteadas.
- Cubículo: El cubículo cuenta con iluminación natural, no se requirió la luz artificial para iluminar el escenario por donde se contará a las personas.

La tabla 5.1 muestra información detallada y descripción de los videos capturados.

Video	Lugar	Tiempo	Frames	FPS	Conteo manual: hacia la Derecha	Conteo manual: hacia la Izquierda
1	Laboratorio	13.75 seg.	375	27	3	2
2	Casa	49.75 seg.	1375	27	3	2
3	Cubículo	66.67 seg.	1838	27	4	4
4	Sala Juntas	89.56 seg.	2475	27	13	14

Tabla 5.1: Descripción de las secuencias de video en la experimentación.

5.2. Prueba con el algoritmo de seguimiento con el contorno de mayor área.

El experimento consistió en proporcionar al sistema contador de personas los archivos de video de prueba. Se verificó la exactitud con que se realizó la cuenta de personas en cada video. La exactitud se define por la ecuación 5.1.

$$S_E = \frac{PCS}{PCM} \tag{5.1}$$

Donde:

- PCS se refiere al número total de personas que contó el sistema en ambos sentidos.
- PCM es el número de personas que manualmente se contaron en la secuencia de video de prueba.
- S_E se refiere a la exactitud del sistema

La tabla 5.2 muestra información detallada y descripción de los videos analizados mediante el sistema de visión computacional para el conteo de personas, con el algoritmo de seguimiento por el contorno de mayor área.

Video	Conteo Sistema: hacia la Derecha	Conteo Sistema: hacia la Izquierda	S_E. Derecha	S_E. Izquierda	S_E. General
1	3	2	1.0	1.0	1.0
2	3	2	1.0	1.0	1.0
3	4	4	1.0	1.0	1.0
4	13	13	1.0	0.928	0.962

Tabla 5.2: Descripción de las secuencias de video en la experimentación, con el algoritmo de seguimiento por el contorno de mayor área.

5.2.1. Resultados con el algoritmo de seguimiento por la envolvente de contornos.

La tabla 5.3 muestra información detallada y descripción de los videos analizados mediante el sistema de visión computacional para el conteo de personas, algoritmo de seguimiento por la envolvente de contornos.

Video	Conteo Sistema: hacia la Derecha	Conteo Sistema: hacia la Izquierda	S_E. Derecha	S_E. Izquierda	S_E. General
1	3	2	1.0	1.0	1.0
2	3	2	1.0	1.0	1.0
3	4	4	1.0	1.0	1.0
4	13	12	1.0	0.857	0.925

Tabla 5.3: Descripción de las secuencias de video en la experimentación, con algoritmo de seguimiento por la envolvente de contornos.

5.2.2. Resultados con el seguimiento de video con el algoritmo de Meanshift.

La tabla 5.4 muestra información detallada y descripción de los videos analizados mediante el sistema de visión computacional para el conteo de personas, con seguimiento

de video con el algoritmo de Meanshift.

Video	Conteo Sistema: hacia la Derecha	Conteo Sistema: hacia la Izquierda	S_E. Derecha	S_E. Izquierda	S_E. General
1	2	2	0.666	1.0	0.8
2	2	1	0.666	0.5	0.6
3	4	3	1.0	0.75	0.875
4	7	3	0538	0.214	0.370

Tabla 5.4: Descripción de las secuencias de video en la experimentación, con seguimiento de video con el algoritmo de Meanshift.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones

Se implementó un sistema de visión computacional para el conteo de personas, implementando tres algoritmos distintos para el seguimiento de los objetos identificados, los cuales son:

- Contorno de mayor área
- Envolvente de contornos
- seguimiento por el algoritmo de Meanshift

En las pruebas realizadas el algoritmo de contorno de mayor área fue el que obtuvo los mejores resultados, al obtener el menor numero de errores en el conteo de personas en ambas direcciones. La heurística de la envolvente de mayor área obtuvo el segundo mejor desempeño en la actividad realizada por el sistema, al tener pocos errores; y finalmente el algoritmo de Meanshift comportándose de una manera muy ineficiente con muchos errores en sus resultados, atribuible a la difícil situación de sintonizar los parámetros del algoritmo.

Bajo condiciones controladas, se concluye que los tres algoritmos llegan a tener buen desempeño, algunas de estas condiciones son: espacios bien iluminados, poca o nula variación de la iluminación, y acceso de una sola persona a la vez.

En el video de prueba número 4 se observan problemas de oclusión y multiobjetos en la escena, provocando un mal desempeño de los algoritmos, sin embargo las heurísticas propuestas se comportan en forma robusta ante la problemática encontrada.

Es posible implementar un sistema de visión computacional para el conteo de personas de bajo costo, los algoritmos presentados no exigen el uso de equipos de computó costosos y la cámara de video para la captura es una cámara web económica.

La implementación de estos sistemas sirve como material didáctico en la materia de visión computacional, facilitando el rápido desarrollo e implementación de los proyectos.

6.2. Trabajos futuros

Algunos de los trabajo futuros que se proponen son:

- Hacer que el sistema sea multi-objeto, es decir, que pueda realizar el seguimiento no solamente de una persona.
- Atacar el problema de oclusión en el seguimiento de las personas identificadas.
- Implementar en un sistema embebido el sistema de visión con la finalidad de hacerlo portátil.
- Identificar por medio de plantillas la cabeza de una persona.
- Realizar la identificación de personas, colocando la cámara en un ángulo diferente al actualmente establecido; generar una base de datos de los rostros para lograr el reconocimiento de personas.

Apéndice A

Código fuente del algoritmo de seguimiento por el contorno de mayor área.

```
import numpy as np
import cv2
import time

cap = cv2.VideoCapture("laboratorio")

fgbg = cv2.BackgroundSubtractorMOG()
kernel = np.ones((5,5),np.uint8)

ret, primerframe = cap.read()
time.sleep(1)

ret, primerframe = cap.read()
grises = cv2.cvtColor(primerframe,cv2.COLOR_BGR2GRAY)
primerth3 = cv2.adaptiveThreshold(grises,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,2)

lista = []
bandera = 0
der = 0
izq = 0
ij = 0

while(cap.isOpened()):
    ret, frame = cap.read()
    grises = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    th3 = cv2.adaptiveThreshold(grises,255,\
    cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,2)
```

```
diferencia = primerth3 - th3
contours, hierarchy = cv2.findContours(diferencia,cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE)
cont_aux = contours[0]

for cont in contours:
    if cv2.contourArea(cont) > 800:
            if cv2.contourArea(cont) > cv2.contourArea(cont_aux):
                    cont_aux = cont

cv2.drawContours(frame, cont_aux, -1, (0,255,0), 1)
x,y,w,h = cv2.boundingRect(cont_aux)
cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)

diferencia = cv2.cvtColor(diferencia,cv2.COLOR_GRAY2RGB)
cv2.drawContours(diferencia, cont_aux, -1, (0,255,255), 2)
cv2.rectangle(diferencia,(x,y),(x+w,y+h),(0,255,0),3)

recorte = frame[y:y+h, x:x+w]

if cv2.contourArea(cont_aux) > 800:
    cv2.line(frame,(320,0),(320,480),(255,0,0),2)
    lista.append([x,y])
    bandera = 1
else:
    bandera = bandera + 1
    maximo = 0
    minimo = 0

    if bandera == 2:
       for x in lista:
          if x[0] > maximo:
             maximo = x[0]

       minimo = 600
       for x in lista:
          if x[0] < minimo:
             minimo = x[0]

       resultado = maximo - minimo

       if resultado < 80:
            continue

       direccion = 0;
       for i in range(0, len(lista)-1):
          direccion = direccion + lista[i][0] - lista[i+1][0]
          ij = ij + 1

       if ij < 4:
            ij = 0
            continue

       ij = 0

       if direccion < 0:
          if lista[0][0] < 200:
             der = der + 1
```

```
            elif direccion > 0:
                if lista[0][0] > 300:
                    izq = izq + 1

            print '\nderecha', der
            print '\nizquierda', izq
        lista = []

    cv2.line(frame,(320,1),(320,480),(0,255,255),2)

    cv2.imshow('frame o',frame)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Apéndice B

Código fuente del algoritmo de seguimiento por la envolvente de contornos.

```
import numpy as np
import cv2
import time

cap = cv2.VideoCapture("casa.avi")
fgbg = cv2.BackgroundSubtractorMOG()
kernel = np.ones((5,5),np.uint8)
ret, primerframe = cap.read()
time.sleep(1)

ret, primerframe = cap.read()
grises = cv2.cvtColor(primerframe,cv2.COLOR_BGR2GRAY)
primerth3 = cv2.adaptiveThreshold(grises,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,2)

lista = []
bandera = 0
der = 0
izq = 0
ij = 0

while(cap.isOpened()):

        ret, frame = cap.read()
        grises = frame

        if ret == True:
                grises = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
```

```
th3 = cv2.adaptiveThreshold(grises,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,2)

diferencia = primerth3 - th3
contours, hierarchy = cv2.findContours(diferencia,cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE)

min_x = 0
min_y = 0
max_x = 0
max_y = 0

for cont in contours:
        if cv2.contourArea(cont) > 300:
                min_x = 1000
                min_y = 1000
                max_x = 0
                max_y = 0
                break

for cont in contours:
        if cv2.contourArea(cont) > 300:
                x,y,w,h = cv2.boundingRect(cont)

                if min_x > x:
                        min_x = x

                if min_y > y:
                        min_y = y

                if max_x < x+w:
                        max_x = x+w

                if max_y < y+h:
                        max_y = y+h

x = min_x
y = min_y
w = max_x - min_x
h = max_y - min_y

cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
recorte = frame[y:y+h, x:x+w]

if w > 50 or h > 50:
        cv2.line(frame,(320,0),(320,480),(255,0,0),2)
        lista.append([x,y])
        bandera = 1
else:
        bandera = bandera + 1
        maximo = 0
        minimo = 0

        if bandera == 2:
                for x in lista:
                        if x[0] > maximo:
                                maximo = x[0]
                minimo = 600
                for x in lista:
```

```
                                        if x[0] < minimo:
                                                minimo = x[0]

                                resultado = maximo - minimo

                                if resultado < 80:
                                        continue

                                direccion = 0;
                                for i in range(0, len(lista)-1):
                                        direccion = direccion + lista[i][0] - lista[i+1][0]
                                        ij = ij + 1

                                if ij < 4:
                                        ij = 0
                                        continue

                                ij = 0

                                if direccion < 0:
                                        if lista[0][0] < 200:
                                                der = der + 1
                                elif direccion > 0:
                                        if lista[0][0] > 300:
                                                izq = izq + 1

                                print '\nderecha', der
                                print '\nizquierda', izq
                        lista = []

                cv2.line(frame,(320,1),(320,480),(0,255,255),2)

                cv2.imshow('frame o',frame)
                k = cv2.waitKey(30) & 0xff
                if k == 27:
                        break
        else:
                break

cap.release()
cv2.destroyAllWindows()
```

Apéndice C

Código fuente del seguimiento de video con el algoritmo de Meanshift.

```
import numpy as np
import cv2
import time

cap = cv2.VideoCapture("cubiculo.avi")
fgbg = cv2.BackgroundSubtractorMOG()
kernel = np.ones((5,5),np.uint8)
ret, primerframe = cap.read()
time.sleep(1)

ret, primerframe = cap.read()
grises = cv2.cvtColor(primerframe,cv2.COLOR_BGR2GRAY)
primerth3 = cv2.adaptiveThreshold(grises,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,2)

lista = []
bandera = 0
der = 0
izq = 0
ij = 0

while(cap.isOpened()):

    ret, frame = cap.read()

    grises = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    th3 = cv2.adaptiveThreshold(grises,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,2)
    diferencia = primerth3 - th3
```

```
contours, hierarchy = cv2.findContours(diferencia,cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE)
cont_aux = contours[0]

for cont in contours:
    if cv2.contourArea(cont) > 80:
        if cv2.contourArea(cont) > cv2.contourArea(cont_aux):
            cont_aux = cont

cv2.drawContours(frame, cont_aux, -1, (0,255,0), 1)

c,r,w,h = cv2.boundingRect(cont_aux)
track_window = (c,r,w,h)

roi = frame[r:r+h, c:c+w]
hsv_roi =  cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
mask = cv2.inRange(hsv_roi, np.array((50., 100.,32.)), np.array((180.,180.,255.)))
roi_hist = cv2.calcHist([hsv_roi],[0],mask,[180],[0,180])
cv2.normalize(roi_hist,roi_hist,0,255,cv2.NORM_MINMAX)
term_crit = ( cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 300, 2)

if cv2.contourArea(cont_aux) > 700:

    while(1):
        ret ,frame = cap.read()
        grises = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        th3 = cv2.adaptiveThreshold(grises,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY_INV,11,2)
        diferencia = primerth3 - th3
        contours, hierarchy = cv2.findContours(diferencia,cv2.RETR_TREE,cv2.CHAIN_APPROX_NONE)
        cont_aux = contours[0]

        for cont in contours:
            if cv2.contourArea(cont) > 80:
                    if cv2.contourArea(cont) > cv2.contourArea(cont_aux):
                            cont_aux = cont

        if cv2.contourArea(cont_aux) > 0:

                p = 1
                if ret == True:
                    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
                    dst = cv2.calcBackProject([hsv],[0],roi_hist,[0,180],1)
                    ret, track_window = cv2.meanShift(dst, track_window, term_crit)

                    x,y,w,h = track_window
                    img2 = cv2.rectangle(frame, (x,y), (x+w,y+h),(0,0,255),2)
                    cv2.imshow('img', frame)

                    k = cv2.waitKey(60) & 0xff
                    if k == 27:
                        break
                    else:
                        cv2.imwrite(chr(k)+".jpg",img2)

                else:
                    break

        else:
```

```
                p = 0

            if p == 1:
                    cv2.line(frame,(320,0),(320,480),(255,0,0),2)
                    lista.append([x,y])
                    bandera = 1
            else:
                bandera = bandera + 1
                maximo = 0
                minimo = 0

                if bandera == 2:
                        for x in lista:
                                if x[0] > maximo:
                                        maximo = x[0]

                        minimo = 600
                        for x in lista:
                                if x[0] < minimo:
                                        minimo = x[0]

                        resultado = maximo - minimo

                        if resultado < 80:
                                continue

                        direccion = 0;
                        for i in range(0, len(lista)-1):
                                direccion = direccion + lista[i][0] - lista[i+1][0]
                                ij = ij + 1

                        if ij < 4:
                                ij = 0
                                continue
                        ij = 0

                        if direccion < 0:
                                if lista[0][0] < 200:
                                        der = der + 1
                        elif direccion > 0:
                                if lista[0][0] > 300:
                                        izq = izq + 1

                        print '\nderecha', der
                        print '\nizquierda', izq
                lista = []
                break

    cv2.line(frame,(320,1),(320,480),(0,255,255),2)

    cv2.imshow('frame o',frame)
    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Apéndice D

Código fuente para obtener los FPS de los videos capturados.

```
import numpy as np
import cv2
from time import time

cap = cv2.VideoCapture("sala_juntas.avi")

cuadro = 0

start_time = time()

while(1):
        ret, frame = cap.read()
        if ret == True:
                cuadro = cuadro + 1
                cv2.imshow('frame o',frame)
                k = cv2.waitKey(30) & 0xff
                if k == 27:
                        break
        else:
                break

elapsed_time = time() - start_time

print("Elapsed time: %0.10f seconds." %elapsed_time)
print "Frames: ", cuadro
print 'FPS: ', cuadro / elapsed_time

cap.release()
cv2.destroyAllWindows()
```

Referencias

[Arévalo04] Arévalo, V., González, J., y Ambrosio, G. La librería de visión artificial opencv. aplicación a la docencia e investigación. *Base Informática*, 40:61–66, 2004.

[Ban15] Tasa de crecimiento de la población, 2015. Consultado el 28 de Agosto de 2015.
URL `http://www.worldbank.org/depweb/spanish/modules/social/pgr/chart1.html`

[blu14] B-top the people counter solution by blue eye video, 2014. Consultado el 14 de Octubre de 2015.
URL `http://www.blueeyevideo.com/PeopleCounting.php`

[Bradski08] Bradski, G. y Kaehler, A. *Learning OpenCV: Computer vision with the OpenCV library*. .ºReilly Media, Inc.", 2008.

[Brookner98] Brookner, E. *Tracking and Kalman filtering made easy*. Wiley New York, 1998.

[Cortés S.13] Cortés S., D. F. Implementación de un algoritmo para el reconocimiento y análisis de peatones utilizando visión por computador. 2013.

[Cuevas13] Cuevas, J. T. R. y Valdez, M. G. M. Conteo de personas mediante video-camaras. 2013.

[DelaFuente12] De la Fuente, J. A., Felipe-Riverón, E., y Garduño-Calderón, E. Segmentación del disco óptico en imágenes de retina mediante la transformada de

hough y los contornos activos. *Research in Computer Science*, 58:117–131, 2012.

[Dep11] Tecnicas de segmentacion de imagenes, 2011. Consultado el 13 de Octubre de 2015.
URL `http://www.elai.upm.es/moodle/mod/resource/view.php?id=452`

[Duque11] Duque, R. G. Python para todos. 2011.

[fra15] Cómo utilizar los fotogramas en vídeo, 2015. Consultado el 13 de Octubre de 2015.
URL `http://www.casanovafoto.com/blog/2014/10/frame-rate/`

[Fukunaga75] Fukunaga, K. y Hostetler, L. D. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32–40, 1975.

[García12] García, J., Gardel, A., Bravo, I., Lázaro, J. L., Martínez, M., y Rodríguez, D. Detección y seguimiento de personas basado en estereovisión y filtro de kalman. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 9(4):453–461, 2012.

[Gibson14] Gibson, J. J. *The Ecological Approach to Visual Perception: Classic Edition.* Psychology Press, 2014.

[gui10] Pixeles y resoluciones, lo que hay que saber sobre estos términos. `http://guioteca.com/tecnologia/pixeles-y-resoluciones-lo-que-hay-que-saber-sobre-estos-terminos`, 2010. Consultado el 13 de Octubre de 2015.

[HIS15] Hispalinux, 2015. Consultado el 13 de Octubre de 2015.
URL `http://hispalinux.es/`

[Hjelmås01] Hjelmås, E. y Low, B. K. Face detection: A survey. *Computer vision and image understanding*, 83(3):236–274, 2001.

[Kim02] Kim, J.-W., Choi, K.-S., Choi, B.-D., y Ko, S.-J. Real-time vision-based people counting system for the security door. *En International Technical Conference on Circuits/Systems Computers and Communications*, págs. 1416–1419. 2002.

[Laganière11] Laganière, R. *OpenCV 2 Computer Vision Application Programming Cookbook: Over 50 recipes to master this library of programming functions for real-time computer vision.* Packt Publishing Ltd, 2011.

[Marr82a] Marr, D. Vision san francisco. *W. H. Freeman and Company*, 1982.

[Marr82b] Marr, D. y Vision, A. A computational investigation into the human representation and processing of visual information. *WH San Francisco: Freeman and Company*, 1982.

[Masoud01] Masoud, O. y Papanikolopoulos, N. P. A novel method for tracking and counting pedestrians in real-time using a single camera. *Vehicular Technology, IEEE Transactions on*, 50(5):1267–1278, 2001.

[Mera G.15] Mera G., J. I. *Estudio de los efectos de perspectiva en contadores de personas basados en vídeo con lentes de gran angular.* Tesis Doctoral, 2015.

[Muy15] Eadweard muybridge, 2015. Consultado el 12 de Octubre de 2015.
URL `https://es.wikipedia.org/wiki/Eadweard_Muybridge`

[Num15] Numpy, 2015. Consultado el 13 de Octubre de 2015.
URL `https://es.wikipedia.org/wiki/NumPy`

[Ope15] Opencv (open sourse computer vision), 2015. Consultado el 01 de Octubre de 2015.
URL `http://opencv.org/`

[ori10] Modelo de un taller: Orígenes del cine, 2010. Consultado el 12 de Octubre de 2015.
URL `https://tallerelmate.wordpress.com/2010/12/01/modelo-de-un-taller-origenes-del-cine-i/`

[Patil04] Patil, R., Rybski, P. E., Kanade, T., y Veloso, M. M. People detection and tracking in high resolution panoramic video mosaic. *En Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, tomo 2, págs. 1323–1328. IEEE, 2004.

[Reyes15] Reyes, H. O. Robot mÖvil autÓnomo seguidor de lÍnea con raspberry pi y librerÍa de visiÓn computacional (opencv). 2015.

[Septian06] Septian, H., Tao, J., y Tan, Y.-P. People counting by video segmentation and tracking. *En Control, Automation, Robotics and Vision, 2006. ICARCV'06. 9th International Conference on*, págs. 1–4. IEEE, 2006.

[Sucar11] Sucar, L. E. y Gómez, G. Visión computacional. *Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, México*, 2011.

[tea15] Cuál es la mejor cámara para grabar tu curso, 2015. Consultado el 12 de Octubre de 2015.
URL `https://blog.teachlr.com/mejor-camara-para-grabar-tu-curso/`

[Wikipedia15] Wikipedia. Modos de video: 480p, 2015.
URL `https://es.wikipedia.org/wiki/480p`

[Yang02] Yang, M.-H., Kriegman, D. J., y Ahuja, N. Detecting faces in images: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(1):34–58, 2002.

[Yang03] Yang, D. B., González-Baños, H. H., y Guibas, L. J. Counting people in crowds with a real-time network of simple image sensors. *En Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, págs. 122–129. IEEE, 2003.

[Yilmaz06] Yilmaz, A., Javed, O., y Shah, M. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.

[Zhao02] Zhao, T. y Nevatia, R. Stochastic human segmentation from a static camera. *En Motion and Video Computing, 2002. Proceedings. Workshop on*, págs. 9–14. IEEE, 2002.

[Zhao09] Zhao, X., Delleandrea, E., y Chen, L. A people counting system based on face detection and tracking in a video. *En Advanced Video and Signal Based Surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*, págs. 67–72. IEEE, 2009.