Universidad Autónoma Metropolitana

Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Ingeniería en Computación

Proyecto Terminal

Seguimiento de personas en un entorno de múltiples cámaras.

Alumno:
Aguayo Ruiz César - 205300512

Fecha de entrega:
06 / Enero / 2011

Trimestre Lectivo:
11-O

Asesores:
M. en C. Arturo Zúñiga López - 28779
Dr. Carlos Avilés Cruz - 24935

.

A mi madre y a mi abuela por apoyarme siempre.

Agradecimientos

A mis asesores: M. en C. Arturo Zuñiga López y al Dr. Carlos Avilés Cruz que dedicaron tiempo y esfuerzo por hacerme mejorar y terminar este proyecto.

A mi madre que ha dado tanto para que logrará esto. Gracias por los sacrificios que hiciste y gracias por apoyarme este último año.

A mi abuela por toda la ayuda no solo a mi sino a toda la familia. Desde que me acuerdo siempre has estado para nosotros Angela.

A mi hermano que a su manera también me apoyaba y me motivó a mejorar en muchos aspectos.

A mis amigos Bernardo, Alfredo, Juan, David (Guille), Vianey,..., a todos aquellos con los que tuve la oportunidad de aprender y divertirme a su lado y todavía me siguen enseñando tantas cosas.

A todos mis profesores de la Universidad Autónoma Metropolitana a lo largo de estos años de aprendizaje, gracias a ellos hoy es posible todo esto.

Muchas gracias a todos.

César Aguayo Ruiz.

Índice

1. Introducción

1.1. Visión por computadora

La visión por computadora[13] es la transformación de la información proveniente de un fotograma o cámara de vídeo ó ambas a una nueva representación. Todas estas transformaciones se realizan para el logro de algún objetivo en particular. Debido a que los humanos somos creaturas visuales la tarea podría parecer muy sencilla. El cerebro humano divide la información visual en varios canales que posteriormente será procesada.

El cerebro tiene un sistema de atención que identifica, en una tarea de manera dependiente, partes importantes de la imagen para examinar, mientras suprime otras áreas que no son importantes. Existe información masiva en la investigación del proceso de visión, que todavía no está completamente entendida.

El ciclo de información en el cerebro regresa a todos los escenarios de procesamiento, incluyendo el hardware (los ojos), que mecánicamente controla la luz vía el iris y enfoca la recepción en la superficie de la retina.

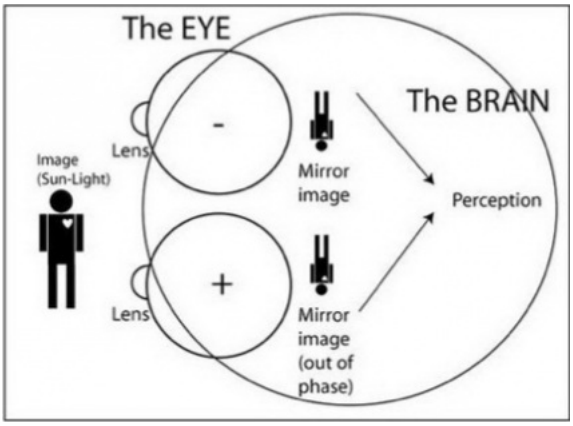


Figura 1: Cómo trabaja el ojo humano

En una máquina de sistema de visión, la computadora recibe una red de números de la cámara ó del disco y eso es todo. Para la gran parte de estas, no existe una contrucción de pátrones para reconocimiento, ni control automático ó enfoque ó apertura, no hay una asociación cruzada por años de experiencia. En efecto, el problema es peor que difícil, ya que además, es dada una vista de espacio dos dimensional (2D) de un mundo 3D.

Cabe mencionar que los datos son corruptos por el ruido y la distorción. Tal corrupción proviene de variaciones en el mundo (clima, luz, reflexión, movimiento), imperfecciones en el lente ó en las

especificaciones mecánicas, integración finita de tiempo en el sensor (motion blur)[1],ruido eléctrico en el sensor ó en otros aparatos electrónicos, y compresión después de la captura de las imágenes, son algunos de los retos a resolver durante el proceso.

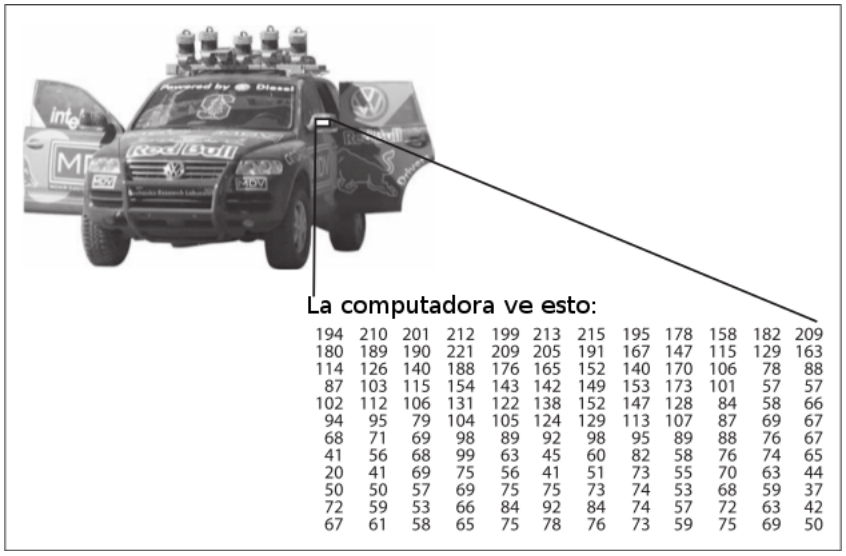


Figura 2: Cómo trabaja la visión por computadora

En el diseño de un sistema, el contexto adicional puede ser frecuentemente usado para trabajar alrededor de las limitaciones impuestas sobre nosotros por el sistema de sensores.

1.2. OpenCV

OpenCV es una librería *open source* escrita en C y C++ y corre sobre Linux,Windows y Mac OS X. Es diseñado para eficiencia computacional y con un fuerte enfoque en aplicaciones de tiempo real. OpenCV también es optimizada para tomar ventaja de múltiples procesadores.

Uno de los objetivos de la librería es proveer el simple uso de una infraestructura de visión por computadora para ayudar a la gente a construir sofisticadas aplicaciones de visión rápidamente. OpenCV contiene más de 500 funciones que abarcan muchas áreas de la visión incluyendo inspección de producto factorial, imágenes médicas, seguridad, interfaz de usuario, calibración de cámaras, estéreo visión, y róbotica.

[1]Desenfoque de movimiento: Producido en la imagen por el movimiento a gran velocidad de los objetos

1.3. Tracking

Cuando lidiamos con una fuente de vídeo, al contrario de imágenes individuales, regularmente se tienen objetivos particulares u objetos que quisieramos seguir a lo largo del campo de visión [2], eso es el *tracking*.

Para realizar esto se necesita pasar un proceso que requiere múltiples funciones antes de llegar a tener un ciclo de imágenes útiles para conseguir una identificación y seguimiento de un objeto a lo largo de la secuencia.

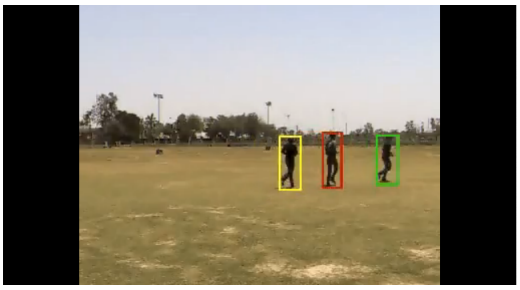


Figura 3: Ejemplo de Tracking

Existen dos tipos de tracking *outdoor*[3] e *indoor*[4], y de él depende el tipo de trabajo que se tendrá que realizar para el procesamiento del vídeo.

Se propone un algoritmo sencillo, capaz de trabajar en condiciones outdoor, ya que el entorno no está completamente controlada; que identifique el fondo de la escena, el primer plano, y los posibles objetos de interés dentro del mismo y su posterior seguimiento, en un entorno de dos cámaras.

El trabajo realizado es *offline*, lo cuál significa que es tomado de grabaciones de vídeo, es decir, no se realiza en tiempo real; además se debe mencionar que en las pruebas no se tiene ningún control sobre las cámaras ni la calidad de las mismas, por lo que es una condición extra a considerar al momento de trabajar con el flujo de imágenes.

Los vídeos empleados fueron proporcionados por el departamento de electrónica de la Universidad Autónoma Metropolitana, unidad Azcapotzalco, ubicadas en el eficio G tercer piso, de la marca AXIS, modelo 206, las condiciones sobre las cuales se trabajaron no pudieron ser totalmente controladas y los problemas a tratar se listan a continuación:

- Cambios de luz. En el campo de visión de la escena, existen cambios de luz debido a las filtraciones en las entradas del pasillo y las escaleras.

[2]Se define como la distancia a lo largo, alto y profundidad, que abarca la vista
[3]Se traduce como exterior, en el cuál no existe un entorno controlado tanto de luz como de ruido
[4]Se traduce como interior, en dónde el entorno es totalmente controlado y sujeto a varias condiciones

- Sombra. Provocada por la iluminación y aumentada por la colocación de las cámaras.
- Reflejos. Provocadas por las paredes blancas y el tipo de luz.
- Ruido. Al no tener control del entorno ni las cámaras, existe ruido debido a la calidad de las imágenes.
- Códecs. Lás cámaras trabajan con diferentes codificadores de audio y vídeo para empaquetar los archivos de vídeo.
- Ángulo de la cámara. El ángulo de la cámara también influye en el proceso ya que genera un cambio en el tamaño de cada blob encontrado.
- Oclusión. Existen demasiadas personas en la escena haciendo más difícil el proceso de seguimiento cuando ocurre una oclusión.

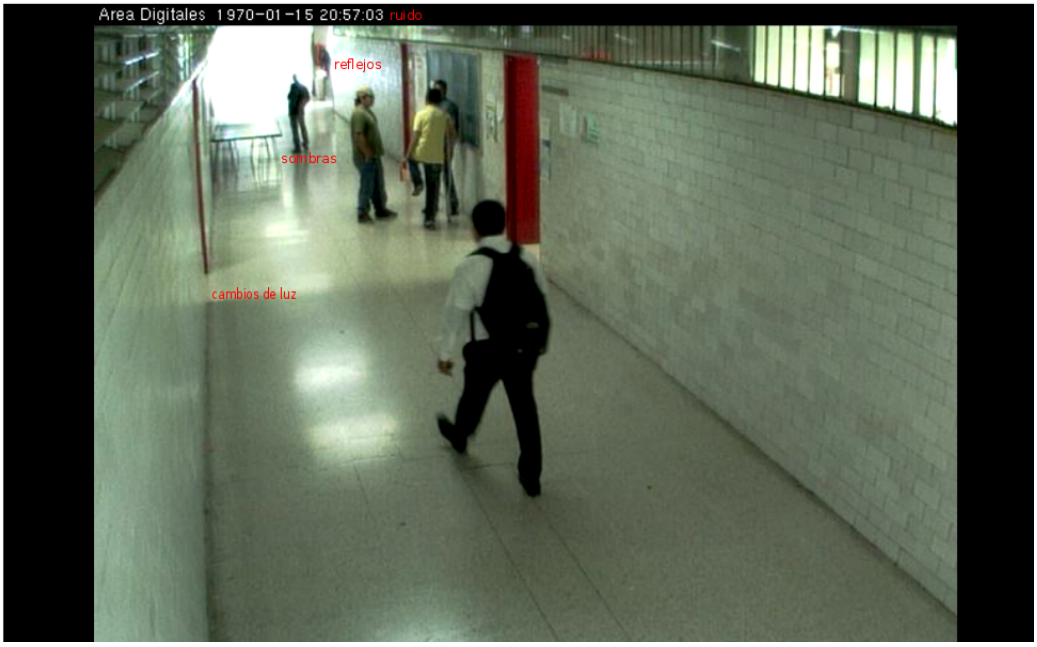


Figura 4: Entorno de trabajo

La solución propuesta se basa en los principios básicos para el tracking, y condicionando el formato del vídeo, el cual será sin códecs para su compresión. Se emplea un modelo Gaussiano basado en [6] para segmentar el fondo y el primer plano, posteriormente se realiza una transformación morfológica para mejorar las zonas de interés y los posibles objetos que se encuentran en movimiento y finalmente a través de cvBlob[4] se realizó la obtención por medio de componentes conectados y trazado de contornos de los blobs en la escena y el seguimiento de los mismos.

2. Estado del Arte

Como otras tecnologías, la detección mediante visión por computadora surge de manera gradual, aparece despúes de un período de investigación y desarrollo en el campo industrial, militar y académico. La aparición de esta tecnología llega acompañada por la madurez en otras tecnologías como las computadoras, las cámaras digitales y la fibra óptica. En el proceso del *tracking* nos interesa entender el movimiento de un objeto, y esta tarea va acompañada de dos componentes principales:

- Identificación. La identificación acumula el encontrar objetos de interés de un frame[5] a otro en el flujo de vídeo.
- Modelado. Ayuda a manejar el hecho de que las técnicas de procesamiento de vídeo, proveen medidas de ruido del objeto u objetos en la posición actual.

El tracking es uno de los principales temas de investigación en escenarios dinámicos, actualmente existen diversos algoritmos para realizar tracking, muchos de ellos trabajan con funciones implementadas en la líbreria de OpenCV, a continuación se muestran algunos trabajos que realizan tracking multicámara:

2.1. Multi-Camera Multi-Person 3D Space Tracking with MCMC in Surveillance Scenarios

En [10] presentan un algoritmo de tracking para un número variable de personas 3D en un entorno de múltiples cámaras con una superposición en el campo de visión. El proceso de seguimiento de múltiples objetos es posicionado en un entorno bayesiano y se basa en un conjunto de estados de múltiples objetos en el espacio con estados de objetos individuales definidos en un mundo 3D.

El método de salto reversible de las cadenas de Markov Monte Carlo (RJ-MCMC) es usado para hacer más eficiente la búsqueda de estado-espacio y recursivamente estimar la configuración de los múltiples objetos.

2.2. Tracking People using Multiple Cameras

CVLAB[1], realiza proyectos de investigación empleando visión por computadora, en el campo del tracking múlti-cámara desarrollaron un algoritmo llamado *POM* que usa un modelo generativo de sustracción de fondo que estima la posición de la gente en un tiempo individual y posteriormente aplican un método de optimización K-shortest Path (KSP) para enlazar todas las detecciones obtenidas individualmente.

[5]Por conveniencia y su constante aplicación en la rama, se llamará asi a cada cuadro de vídeo extraído de una fuente de vídeo

El algoritmo es robusto y maneja técnicas muy avanzadas que permite localizar a cada persona por separado en cada una de las cámaras que enfocan el área de grabación.

2.3. Tracking Human Motion Using Multiple Cameras

En [12] proponen un algoritmo para tracking de movimiento de humanos en un entorno indoor de secuencias de imágenes en escala de grises obtenidas de múltiples cámaras arregladas. El Modelo de Gauss es aplicado para encontrar las coincidencias más parecidas a objetos humanos en los frames consecutivos tomados de las cámaras montadas en diferentes locaciones.

En los trabajos citados existe una constante para el proceso de seguimiento, el entorno de trabajo es indoor, y las cámaras son parte fundamental en cuanto al desarrollo del mismo. Los tres pueden ser considerados como algoritmos robustos y ampliamente estudiados pero sirvieron para encaminar el proceso de trabajo.

3. Fundamentos Teóricos

Una imagen[14] se refiere a una función bidimensional de intensidad de luz $f(x, y)$, donde x e y representan las coordenadas espaciales y el valor de f en un punto cualquiera (x,y) es proporcional al brillo (o nivel de gris) de la imagen.

Una imagen digital es una imagen $f(x, y)$ que se ha discretizado tanto en coordenadas espaciales como en el brillo. Como se mencionó, una computadora recibe de una imagen una matriz de números como información proveniente de una cámara o vídeo, cuyos índices de fila y columna identifican un punto de la imagen y el valor correspondiente elemento de la matriz indica el nivel de gris en ese punto.

3.1. Resolución

Dependiendo de la cantidad de píxeles que tenga el dispositivo, la imagen poseerá más o menos resolución, la resolución pueden ser 128 x 128, 256 x 256,680 x 480, 800 x 600, 1280 x 720 (HD), 1920 x1080 (FHD).

3.2. Tipo de imagen

Se refiere al tipo de dato usado el cual puede ser en escala de grises, a color, 4-canales(RGB + alpha), y cada canal puede contener uno o varios tipos de números enteros o flotantes para representarla.

3.3. Fundamentos del color

La luz solar atraviesa un prisma de cristal y se descompone en los siete colores básicos del espectro visible. Básicamente, los colores que las personas perciben en un objeto están determinados por la naturaleza de la luz reflejada por el objeto.

3.3.1. Modelos de Color

El objetivo de los modelos de color es facilitar la especificación de los colores de alguna forma estándar. En escencia, un modelo de color es una especificación de un sistema de coordenadas en 3D y un subespacio de dicho sistema donde cada color se representa por un punto.

Muchos modelos de color se usan hoy en día, los más comunes se listan a continuación:

- RGB (rojo,verde,azul)
- CMY (cyan,magenta,amarillo)
- YIQ

3.3.2. El Modelo RGB

Cada color aparece en sus componentes espectrales primarias: rojo, verde, azul. Este modelo está basado en el sistema de coordenadas cartesianas. Los valores RGB están en tres vértices; cyan, magenta y amarillo se sitúan en otros tres vértices, el negro corresponde al origen y el blanco en el vértice más alejado del origen. La escala de grises se extiende desde el negro al blanco a lo largo de la diagonal que une esos dos puntos, y los colores son puntos dentro del tetraedo, definidos a lo largo de la diagonal que une esos puntos.

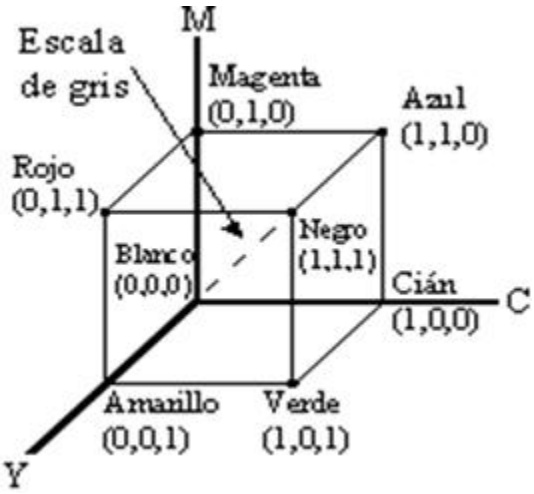


Figura 5: Modelo RGB

3.3.3. Pseudocolor

Se trata de asignar un color a imágenes monocromas basándose en varias propiedades del contenido de nivel de gris.

Viendo una imágen como una función de intensidad 2D, se puede separar la función en dos niveles, donde los valores arriba de un nivel tendrán un valor y los que se encuentran abajo del otro nivel tendrán otro. Suponiendo M planos situados en los niveles $I_1, I_2, ..., I_M$, donde I_0 representa el negro [f(x,y) = 0] e I_L representa blanco [f(x,y) = L].

Considerando que $0 < M < L$, M planos particionan la escala de grises en M+1 regiones y la asignación de color se hace de acuerdo a la relación

$$f(x, y) = c_k \text{ si } f(x, y) \in R_k$$

donde c_k es el color asociado con la k-ésima region R_k entre los niveles I_{k-1},I_k.

Si se quiere asignar color a los niveles de gris en una imagen, se definen dos intervalos de niveles de gris, identificados como i, el intervalo i = 1 para los valores entre 0 y 128 y el intervalo i = 2 para los valores entre 128 y 255. Dado un determinado nivel de gris p, se determina el intervalo i al que pertenece yy se aplica la siguiente fórmula para calcular los tres componentes de color que le corresponde:

$$a = a_{i-1} + \frac{a_i - a_{i-1}}{128}(p - g_{i-1})$$

donde a = RGB, $g_0 = 0$ y $g_1 = 128$

Cabe mencionar que también es posible realizar la operación inversa, es decir, encontrar el equivalente en escala de grises de una imagen RGB.

3.4. Segmentación

La segmentación [15] es el proceso por el cual se extrae de la imagen cierta información subyacente para su posterior uso. La segmentación esta basada en dos principios fundamentales: discontinuidad y similitud. A su vez es importante mencionar la segmentación orientada a bordes (discontinuidad) y orientada a regiones (similitud).

Una región se define como un área de la imagen en la que sus píxeles poseen propiedades similares (de intensidad, color,etc), mientras que un borde es una línea que separa dos regiones, por tanto de diferentes propiedades.

3.5. Transformación de imágenes

Teniendo en mente los datos de entrada, es posible realizar operaciones con los mismos, cada una de estas operaciones tiene un significado,utilidad y aplicaciones específicas.

- Aritméticas: sumar, restar, multiplicar, máximo, etc.
- Booleanas: and, or, not,etc.
- Transformaciones Generales: Transformaciones de histograma, transformaciones de color, Binarización, etc.

3.5.1. Operación píxel a píxel

Una operacion píxel a píxel se puede expresar como una función:

$$R(x, y) = f(A(x, y))$$

Es decir, el valor del píxel resultante es función de, y sólo de, el píxel correspondiente de entrada.

3.5.2. Operaciones de vecindad

Las operaciones de vecindad utilizan el mismo procedimiento excepto que el nuevo valor del píxel en la imagen de salida depende de una combinación de los valores de los píxeles en la vecindad del píxel de la imagen original que está siendo transformada.

Un píxel p de coordenadas (x,y) tiene cuatro vecinos horizontales y verticales cuyas coordenadas vienen dadas por:

$$(x+1, y), (x-1, y), (x, y+1), (x, y-1)$$

Este conjunto de píxeles es denominiado los *4-vecinos* de p. Cada píxel está a una unidad de distancia de (x,y), y algunos de los vecinos de p caen fuera de la imagen digital si (x,y) está en el borde de la imagen.

Los cuatro vecinos en *diagonal* de p tienen las situienes coordenadas:

$$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$$

La union de estos junto a los 4-vecinos, se denominan los *8-vecinos* de p y se representan por $N_8(p)$.

3.5.3. Convolución

Al realizar una convulción, el valor de un píxel depende de la vecindad local de ese píxel. La convolución de dos funciones $f(x)$ y $g(x)$ indicada por $f(x) * g(x)$, se define mediante la integral:

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\alpha) g(x - \alpha) d\alpha$$

Donde α es una variable ficticia para la integración. A veces es conveniente ver la operación de convolución como un producto matricial.

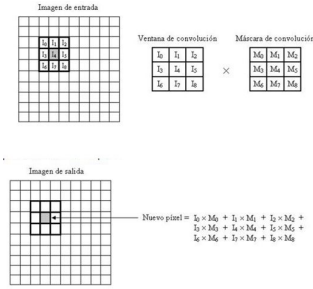


Figura 6: Operación de convolución

3.6. Dilatación

Es la convulción de una imagen o región de una imagen la cual llamaremos A, con algun *kernel*[6] que llamaremos B y el resultado computa el máximo local de los píxeles de B sobre A, dando como resultado que las regiones brillantes sean expandidas y frecuentemente unidas.

$$A \oplus B = \{ \mathbf{d} \in E^2 : d = a + b \text{ para cada } a \in A \text{ y } b \in B\}$$

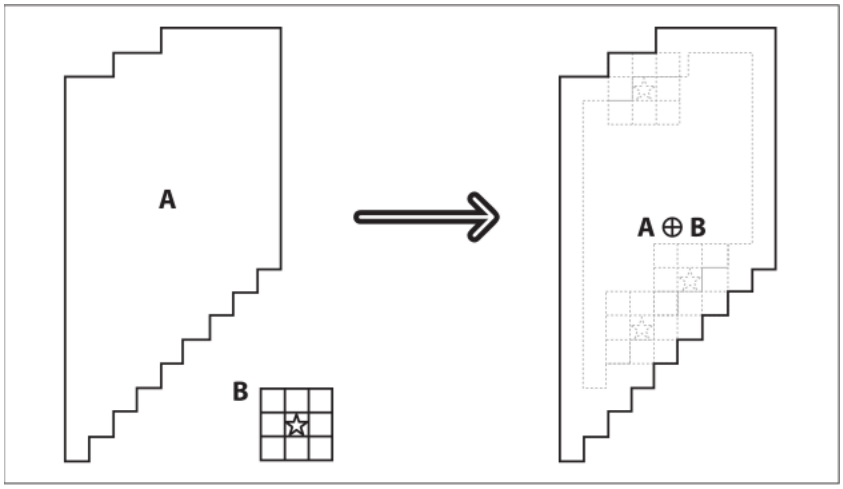


Figura 7: Dilatación Morfológica

Esto es, cuando se aplica esta operación, lo que ocurre es que el valor de algún punto p es colocado como el máximo de todos los puntos que cubre el kernel cuando se alinea en p.

$$dilate(x, y) = \max \text{src}(x + x^{'}, y + y^{'}) \text{ dónde } (x^{'},y^{'}) \in kernel$$

3.7. Erosión

La transformación morfológica de la erosión combina dos conjuntos utilizando la substracción de vectores. Es la operación inversa a la dilatación, ya que trabaja con el mínimo local del kernel sobre la imagen y aisla y reduce las regiones brillantes de la imagen.

$$A \otimes B = \{ \mathbf{d} \in E^2 : d = a + b \text{ para cada } a \in A \text{ y } b \in B\}$$

$$erode(x, y) = \min \text{src}(x + x^{'}, y + y^{'}) \text{ dónde } (x^{'},y^{'}) \in kernel$$

[6]Es un operador el cual puede ser tomado de una plantilla o máscara

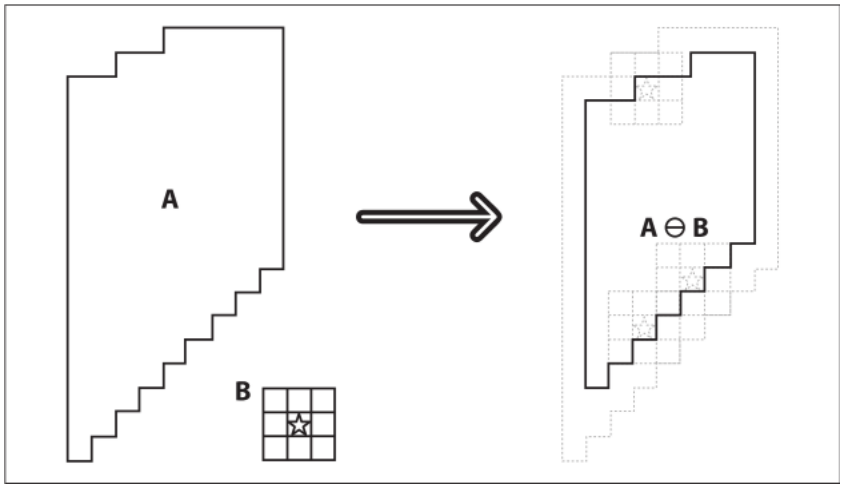


Figura 8: Erosión Morfológica

3.8. Escala de grises

A veces la imagen original no se transforma realmente, sin embargo es posible visualizar la misma imagen bajo diferentes aspectos. Esto se consigue variando lo que se conoce como mapa de niveles de gris. Generalmente los procesadores de imágenes para representar la imagen toman como referencia un mapa de niveles de gris, de tal forma que para la representación de los valores de los píxeles de la imagen original toman como referencia el valor más próximo del mapa de niveles de gris. Con estos valores obtenidos se muestra la imagen en el monitor.

3.9. Histogramas

En el análisis y procesamiento de imágenes, frecuentemente es útil representar lo que se ve como histogramas. El histograma de una imagen es una función discreta que representa el número de píxeles en la imagen en función de los niveles de intensidad, g. La probabilidad P(g) de ocurrencia de un determinado nivel g se define como:

$$P(g) = N(g)/M$$

Donde M es el número de píxeles en la imagen y N(g) es el número de píxeles en el nivel de intensidad g.

3.9.1. Ecualización de un histograma

Sea r una variable que represente los niveles de gris de la imagen a mejorar. En la parte inicial de la presentación se supondrá que los píxeles son cantidades continuas que han sido normalizadas de forma que pertenezcan al intervalo [0,1], con r = 0 representando el negro y r = 1 el blanco.

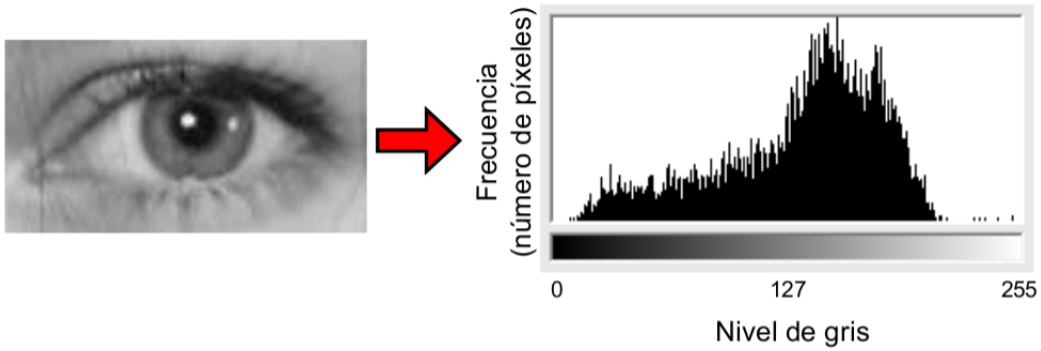


Figura 9: Ejemplo de un Histograma

Posteriormente consideraremos una formulación discreta y permitiremos que los valores varíen en el intervalo [0,L-1].

Para cada r del intervalo [0,1], nos centramos en las transformaciones de la forma

$$s = T(r)$$

que producen un nivel s para cada valor de píxel r de la imagen original. Se supone que la función de transformación de la ecuación anterior verifica las condiciones :

a) T(r) es de valor único y monótonamente creciente en el intervalo $0 \leq r \leq 1$

b) $0 \leq T(r) \leq 1$ para $0 \leq r \leq 1$

El objetivo al ecualizar una imagen, es *"estirar"* el histograma para lograr que la distribución de los colores abarque de manera casi constante todo el rango[7] de colores.

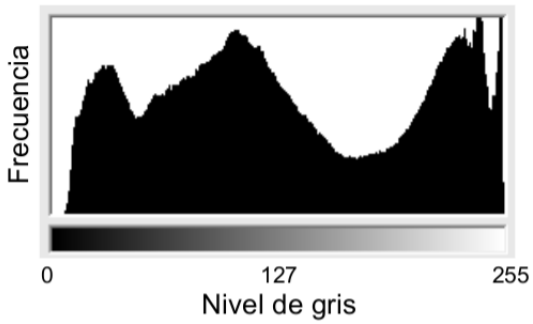


Figura 10: Ecualización de un Histograma

[7]un histograma tiene en teoría 256 celdas

3.10. Contornos

Un contorno es una lista de puntos que representan, en una forma u otra, una curva en la imagen. Esta representación puede ser diferente dependiendo de las circunstancias que la envuelven.

En OpenCV, un contorno se puede representar como una secuencia en la cual cada entrada de la secuencia contiene información acerca de la ubicación del punto siguiente de la curva. Estas secuencias son en si contornos, los cuales, a su vez se pueden representar como poligonos[13].

3.11. Momentos

Una manera de comparar dos contornos es computando sus momentos. Un momento es una característica del contorno computada por la integral (o suma) de todos los puntos que forman el contorno. En general un momento de un contorno puede ser definido de la siguiente manera:

$$m_{p,q} = \sum_{i=0}^{n} I(x,y) x^p \; y^q$$

3.12. Componentes Conectados

El análisis por componentes conectados toma como entrada una máscara, que después es usada para una operación morfológica para eliminar áreas pequeñas de ruido y posteriormente a través de otra transformación, reconstruir el área de los componentes restantes.

Con esto lo que se pretende es encontrar los contornos suficientemente largos que sobrevivieron de los elementos en la escena para posteriormente tratarlos como segmentos.

Figura 11: Ejemplo de la aplicación de Componentes Conectados

4. Desarrollo Funcional

Ya con las herramientas necesarias (ver Apendice), y después de realizar unas pruebas para probar la correcta funcionalidad de las librerías, se procedió al proceso de desarrollo, el cual comprende los siguientes componentes:

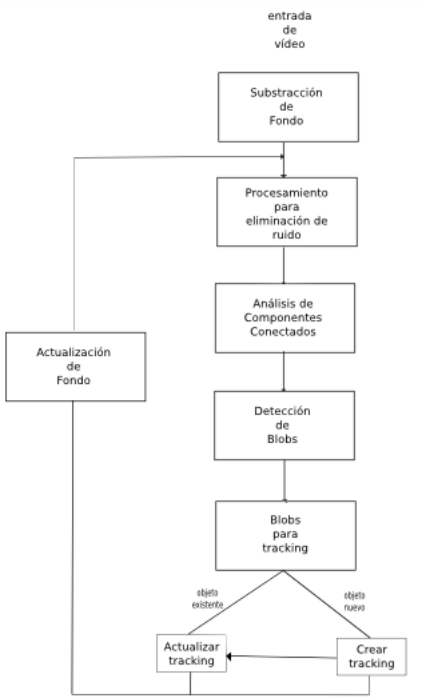


Figura 12: Diagrama de Flujo Principal

4.1. Obtención de Vídeo

Para el primer proceso de este proyecto fue proporcionado un vídeo del pasillo del edificio G 3^{er} piso, el cual viene en un formato llamado asf[8], pero como restricción para el proyecto se trabajará con archivos sin compresión, asi que esto requirió procesamiento extra.

4.1.1. Decodificación con Mencoder

Mencoder[5][9] viene en los repositorios de Debian asi que basta con un *apt-get install mencoder* para tenerlo listo, posteriormente se procede a codificar el vídeo crudo[10] del archivo fuente.

[8] Advanced Streaming Format (o ASF, posteriormente renombrado a Advanced Systems Format) es un formato contenedor de audio y video digital propiedad de Microsoft, diseñado especialmente para el streaming.

[9] Es un codificador de vídeo libre liberado bajo licencia GPL que se incluye en el reproductor multimedia MPlayer.

[10] raw: Video sin compresión.

- mencoder archivo_fuente -ovc raw -o archivo_destino

La opción ovc (output video codec) indica el códec usado para el vídeo de salida. Después de esto tenemos el vídeo listo para el procesamiento.

4.2. Segmentación

Para poder iniciar un seguimiento es importante saber que componentes se encuentran en el frame de vídeo, cuáles son de interés (blob)[11] y cuáles pueden formar parte del fondo y esto se agranda aún más cuándo en el ciclo del mismo existe movimiento incluso de la cámara, a esto se le conoce como Substracción de Fondo.

Se optó por emplear imágenes en escala de grises para buscar reducir el ruido ya que la pruebas en RGB no eran satisfactorias, por lo tanto se trabajará solamente con un canal en los procesos subsecuentes del proyecto. Esto se realiza en el ciclo principal después de capturar cada frame de vídeo se convierte en su equivalente en escala de grises.

```
.
.
.
IplImage *img = cvRetrieveFrame(capture,n);
if(img == NULL)
printf( "Error... Video no encontrado... Terminando");
break;
.
.
.

    if(!gray)
gray = cvCreateImage(cvGetSize(img),IPL_DEPTH_8U,1);
cvCvtColor(img,gray,CV_RGB2GRAY);
```

Comúnmente el *Background* ó fondo de la imagen es considerado como la escena sin la presencia de los objetos de interés, tal como un humano u objetos en movimiento. Se podría decir que Background se compone de objetos sin vida y que permanecen pasivos en la escena. A continuación se presentan algunos métodos estudiados y con los cuales se realizaron pruebas para probar su funcionamiento y decidir cuál trabaja mejor en nuestro caso.

4.2.1. Background Subtraction

Por su simplicidad y porque la localización de la cámara puede estár en muchos contextos *Background Subtraction* ó también conicida como *background differencing* es probablemente la op-

[11] No confundir con Binary Large Object en SQL, en OpenCV un blob se puede definir como un conjunto de píxeles que pueden formar un objeto

eración fundamental en muchas de las aplicaciones de vigilancia. En orden de implementar esta técnica, primero se debe de "aprender" el modelo del fondo.
Una vez hecho lo anterior, se procede a comparar con la imagen actual y las partes del fondo conocido anteriormente son sustraidas. Los objetos restantes después de la substracción son presumiblemente nuevos objetos pertenecientes al primer plano (*foreground*).

Esta técnica funciona considerablemente bien en escenas simples, sufre de una condición que es regularmente violada: que todos los píxeles sean independientes.

4.2.2. Averaging Background Method

Este método básicamente aprende el promedio y la desviación estándar de cada píxel como el modelo del fondo. Para su aplicación se emplea cuatro rutinas de OpenCV

- cvAcc() para acumular imágenes a lo largo del tiempo.
- cvAbsDiff() para acumular frame a frame las diferencias de las imágenes a lo largo del tiempo.
- cvInRange() para segmentar la imagen en fondo y primer plano, una vez que el fondo ha sido aprendido.
- cvOr() para compilar las segmentaciones de diferentes canales de colores en una sola máscara de imagen.

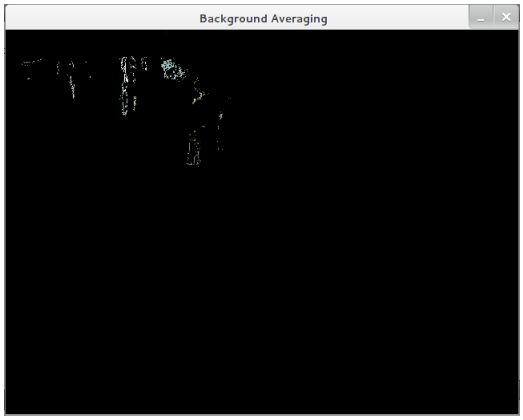


Figura 13: Resultado de Background Averaging Method

Este método no da buenos resultados en nuestra escena puesto que los puntos que se localizan están muy dispersos y no se logra identificar siquiera una forma.

4.2.3. Advanced Background Method (Codebook)

Este método asigna, para cada píxel y cada canal de colores, un valor máximo y uno mínimo y en cada iteración se comprueban estos valores, si se encuentran dentro del rango se consideran como fondo, y para cada cierto número de frames se actualiza el fondo.

Es útil en escenas complicadas, ya que trata objetos que posiblemente son fondo pero se encuentran en movimiento, como árboles, cortinas agitadas por el viento, etc.

Figura 14: Resultado de Codebook Method

Aunque el resultado es mejor que en el método anterior, se sigue perdiendo información en el proceso así que no nos brinda el resultado deseado para la segmentación.

4.2.4. Foreground/Background Segmentation

Basado en [6] se empleará un modelo propuesto para video vigilancia, este último fue el que mejor resultados arrojo en las pruebas y es con el que se pretende continuar para el proceso de tracking.

Regla de Clasificación Para un tipo de objetos de fondo, existe una característica significante que puede ser explotada para separar efectivamente el fondo del primer plano. Sea V_t un vector de valor discreto extraído de una secuencia de imágenes, en el píxel $s = (x, y)$ en el tiempo t.

Descripción del Algoritmo

- Detección de Cambios
  En este paso los píxeles con cambios insignificantes son filtrados por una simple diferencia de fondos. Sea $I(s,t) = I_c(s,t)$ la entrada de la imagen y $B(s,t) = B_c(s,t)$ sea la referencia

del fondo de la imagen mantendia por el sistema en el instante t, y $c \in r, g, b$ representa el componente de color. El fondo y la diferencia temporal está desarrollada como se muestra:

Se obtiene una diferencia de la imagen para cada componente de color con un umbral adaptativo[13]. El resultado de los tres componentes son combinados para generar la diferencia del fondo $F_{bd}(s,t)$ y la diferencia temporal $F_{td}(s,t)$ respectivamente.

- Clasificación de Cambios.
  La diferencia temporal clasifica el cambio de los píxeles en dos tipos:
  * Si $F_{td}(s,t) = 1$ es detectado, el píxel es clasificado con movimiento perteneneciente a un objeto móvil
  * De otro modo, es un píxel estacionario asociado a un objeto estacionario.
- Segmentación de objetos de primer plano
  La probabilidad de que un píxel sea mal etiquetado es muy poca, para los puntos aislados, se realiza una transformación de *opening* y *close* para removerlos. Esto deja a los objetos segmentados en $O(s,t)$.
- Aprendizaje del fondo y mantenimiento El mantenimiento de fondo adapta el modelo de fondo a varios cambios del mismo a través del tiempo, en este método la actualización se realiza en dos étapas:
  * Actualización a cambios graduales del fondo.
  * Actualización de fondo una vez cambiado.

La función cvCreateGaussianBGModel es una implementación basada en una distribución gaussiana para cada píxel en el modelo, considerando la probabilidad de que un píxel tenga un cierto valor en un tiempo t. El resultado de la función devuelve dos imágenes, una para el fondo y otra para el primer plano en un canal.

Asumiendo que el vector V_t es usado para clasificar el píxel s como fondo o primer plano en el tiempo t, las estadísticas de las correspondientes características están actualizadas de la siguiente forma:

$$p_b^{s,t+1} = (1-\alpha_2)p_b^{s,t} + \alpha_2 M_b^{s,t}$$
$$p_v^{s,t+1,i} = (1-\alpha_2)p_v^{s,t,i} + \alpha_2 M_v^{s,t,i}$$
$$p_{vb}^{s,t+1,i} = (1-\alpha_2)p_{vb}^{s,t,i} + \alpha_2 (M_b^{s,t} \wedge M_v^{s,t,i})$$

para i = 1,...,N_2 dónde α_2 es rango de control de aprendizaje de la característica.

Los valores booleanos que determinan las etiquetas son generadas de la siguiente manera:

Si $M_b^{st} = 1$ entonces s es etiquetado como fondo en el tiempo t, de otra manera $M_b^{st} = 0$

El código que emplea esta función se muestra a continuación:

```
.
.
.
CvGaussBGStatModelParams *params = new CvGaussBGStatModelParams;
params → win_size = 1;
params → n_gauss = 5;
params → bg_threshold = 0.7;
params → std_threshold = 3.5;
params → minArea = 15;
params → weight_init = 0.05;
params → variance_init = 30;

//Crear el CvBGStatModel
CvBGStatModel *bgModel = cvCreateGaussianBGModel(frame ,params);
.
.
.
//Después del procesamiento, se actualiza el modelo del fondo

    bgModel→update(frame,bgModel,-1);
```

Figura 15: Resultado de BGstatModel

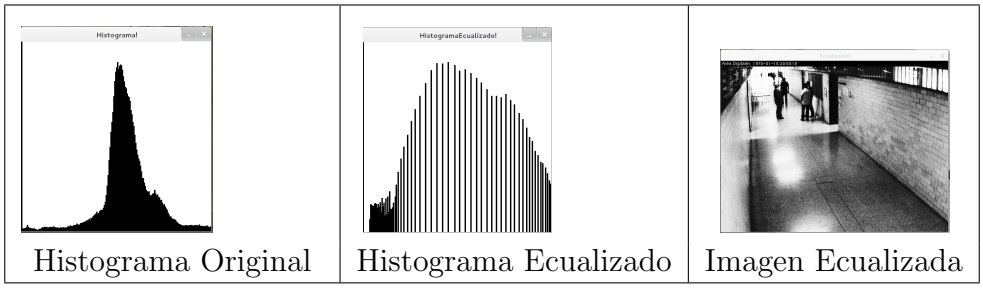
4.3. Procesamiento

Al trabajar con vídeo se debe tener en cuenta que las condiciones que envuelven al mismo pueden no ser las mejores o deseadas para realizar un proceso de visión por computadora, ya que dependiendo de la calidad de la cámara con la cual se toma el vídeo ó el escenario, los problemas resultantes incluyen: ruido, cambios de luz, sombras, etc.

Así después de tener la segmentación del vídeo aún queda mucho con que lidiar y para esto se debe de trabajar con operadores de alto nivel que están definidas en la estructura de las imágenes para cumplir tareas cuyo significado está definido en el contexto gráfico. En nuestro caso, buscarémos mejorar las formas obtenidas del proceso de segmentación y reducir el ruido.

4.3.1. Aplicación de Histograma

Las primeras prebas fueron mediante la ecualización del histograma de la imagen, buscando tener una mejor calidad y reducir las áreas claras ó muy oscuras de la escena,el resultado del histograma de la imagen se muestra a continuación:



Histograma Original | Histograma Ecualizado | Imagen Ecualizada

Figura 16: Aplicación de Histograma

El resultado no fue el óptimo, ya que siguen sin verse definidas las formas sobre las cuales se trabajarán.

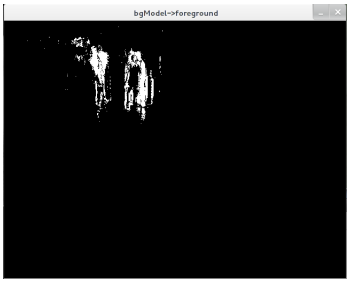


Figura 17: Resultado de la aplicación del Histograma sobre la segmentación

4.3.2. Aplicación de Dilatación

Se procedió a trabajar con transformaciones morfológicas, pretendiendo eliminar las zonas no deseadas en el modelo de fondo y definir las formas de las personas. En este caso se empleará un kernel predeterminado tomado de[13] de 2 x 2 y de forma elíptica, para mantener lo más posible las formas de las figuras.

Para construir el kernel con el que se desea trabajar, OpenCV proporciona la función cvCreateStructingElementEx(), el cual tiene la siguiente estructura:

IplConvKernel cvCreateStructingElementEx(int cols, int rows,int anchor_x, int anchor_y, int shape, int* values = NULL);*

donde:
cols y rows : indican el tamaño del rectángulo que contiene a la matriz.
anchor_x y anchor_y : Son las coordenadas (x,y) del punto de anclaje dentro del rectángulo envolvente del kernel.
shape: es la forma que dará el kernel a la transformación, las diferentes formas se muestran en la siguiente tabla.

Shape value	Meaning
CV_SHAPE_RECT	The kernel is rectangular
CV_SHAPE_CROSS	The kernel is cross shaped
CV_SHAPE_ELLIPSE	The kernel is elliptical
CV_SHAPE_CUSTOM	The kernel is user-defined via values

Figura 18: Tabla de posibles formas del kernel

El resultado del proceso de la dilatación se muestra en la siguiente figura:

Figura 19: Resultado de Dilatación sobre la segmentación

4.3.3. Aplicación de Erosión

De igual manera se aplicó la transformación de erosión en el modelo de fondo y aplicando el mismo kernel para la convolusión, el resultado es el siguiente:

Figura 20: Resultado de Erosión sobre la segmentación

Como se puede observar, los resultados son mucho mejores que con la aplicación del histograma.

4.3.4. Aplicación de Openig con MorphologyEx

Algunas veces estas operaciones no bastan por lo cual OpenCV ofrece la opción de combinarlas y obtener el mejores resultados, o agregar más características, este es el caso de la función cvMorphologyEx que combina las transformaciones básicas en un solo proceso.

Esta función permite realizar varias operaciones, en nuestro caso se aplicará una operación de CV_MOP_OPEN que primero erosiona y luego dilata la imagen para eliminar ruido y mejorar las formas de la entrada de vídeo.

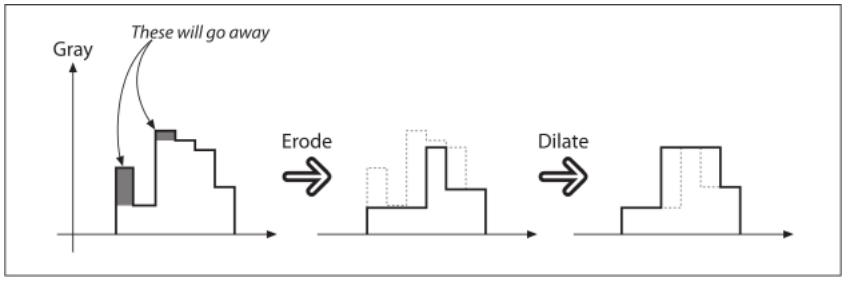


Figura 21: Operación de Open en cvMorphologyEx

El opening es usado frecuentemente para contar las regiones en una imagen binarizada, con esto en mente, el resultado esperado es la correcta identificación de los objetos en la escena.

El resultado del opening en el modelo de fondo se muestra a continuación:

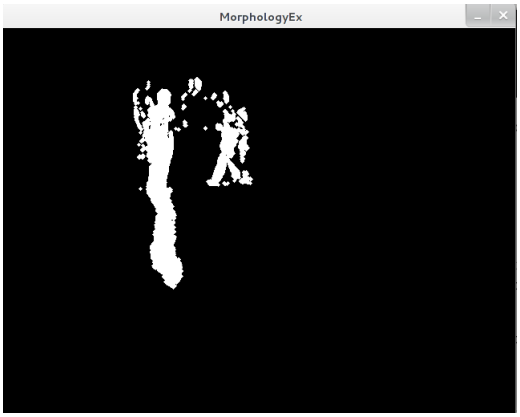


Figura 22: Resultado de CV_MOP_OPEN

4.4. cvBlob

En una imagen binarizada, es decir, tan solo en blanco y negro, y tomando en consideración todo el ruido alrededor de ellos ¿cómo determinar dónde termina un blob y dónde comienza otro?, para este proceso del desarrollo se encontró una librería[4] basada en componentes conectados[13] y que además raliza el tracking de los mismos.

cvBlobs es una librería para visión por computador para detectar regiones conectadas en imágenes binarizadas, emplea un análisis de componentes conectados[7], el cual se basa a grandes rasgos en una aplicación de teorías de gráfos, donde los subconjuntos de componentes conectados son etiquetados de manera heurística.

4.4.1. Adentrandose en cvBlob

El algoritmo de etiquetamiento en cvBlob está basado en[8], en el siguiente diagrama se muestran los 4 procesos principales que se aplican.

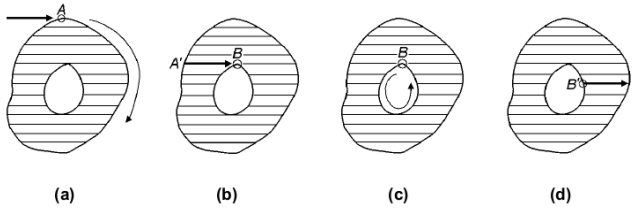


Figura 23: 4 pasos principales en la localización de puntos y etiquetado de componentes.

El proceso es el siguiente, cuando se encuentra un punto de contorno externo por primera vez, en este caso A, se hace una trazo completo hasta regresar al mismo, y se asigna la etiqueta de A y a todos los puntos del contorno.

En la figura 23b, cuando un punto de un contorno externo $A^{'}$ es encontrado, se procede a buscar todos los subsecuentes píxeles negros (si existen) y se asigna a ellos la misma etiqueta $A^{'}$.

En 23c cuando un punto interno de contorno B es encontrado por primera vez, se asigna la misma etiqueta como el contorno externo del mismo componente. Se traza un contorno interno conteniendo B y además asignando a todos los puntos del contorno la misma etiqueta que B.

En la figura 23d, cuando un punto interno de contorno etiquetado, digamos $B^{'}$, es encontrado, se sigue la linea buscando todos los subsecuentes píxeles negros,(si existen), y se asigna a ellos la misma etiqueta $B^{'}$.

Por simplicidad se asume que los píxeles en la columna más alta siempre son blancos(si no lo son, se agrega una columna falsa con píxeles blancos). Para un documento de imagen I, se asocia con I una imagen acompañante L, la cual almacenará la información de las etiquetas. Inicialmente, todos los puntos de L son 0 (es decir, estan sin etiquetar).Posteriormente se escanea la imagen I para encontrar un píxel negro. Por último sea C el indice de las etiquetas para los componentes, inicialmennte C es 1.

Todo lo anterior se puede resumir en 3 pasos lógicos:

1. Se encuentra un nuevo punto externo y todos los que forman ese contorno. Si P está sin etiquetar y el píxel encima de él es un píxel blanco, P debe de ser contorno externo de un nuevo componente. Así que se asigna C a P, mientras tanto se ejecuta el *trazo del contorno* para encontrar el contorno externo, y se asigna la etiqueta C a todos los píxeles del contorno, se incrementa el valor de C en 1.

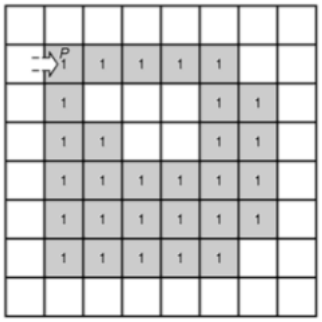


Figura 24: Paso 1 de etiquetamiento cvBlob

2. Si el píxel debajo de P es un píxel blanco sin etiquetar, P debe ser un nuevo contorno interno. Existen 2 posibilidades, la primera P ya está etiquetado (figura 25a), y además es parte de un contorno externo. La segunda es que P esté sin etiquetar (figura 25b), y en este caso el punto predecesor N en la línea de búsqueda (vecino izquiero de P), debe ser etiquetado y posteriormente se asigna a P la misma etiqueta de N. En ambos casos se procede a ejecutar el *trazo de contorno* para encontrar el contorno interno que contiene a P, y se asigna la misma etiqueta a todos los píxeles del contorno.

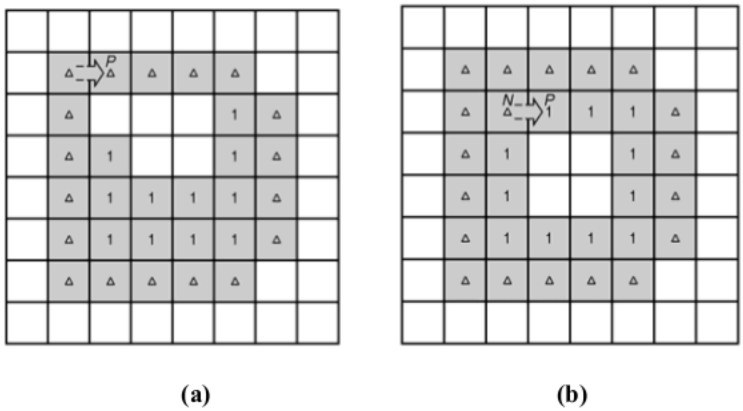


Figura 25: Paso 2 en el proceso de etiquetamiento cvBlob

3. Si P es un punto que no se trató en el paso 1 y 2 (no es un punto de contorno), entonces el píxel vecino izquierdo N de P debe ser etiquetado. La misma etiqueta de N se asigna a P.

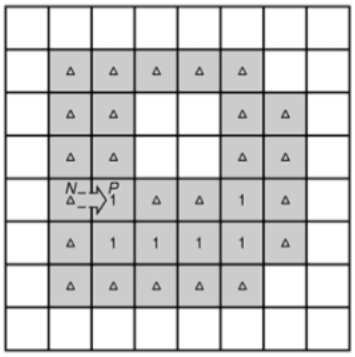


Figura 26: Paso 3 de etiquetamiento cvBlob

En orden de evitar el trazo de contorno para el punto Q, se marcan los píxeles blancos circundantes de un componente con un entero negativo(-). Por lo tanto al realizar la línea de búsqueda para Q, el píxel debajo del mismo ya no es un píxel sin etiquetar.

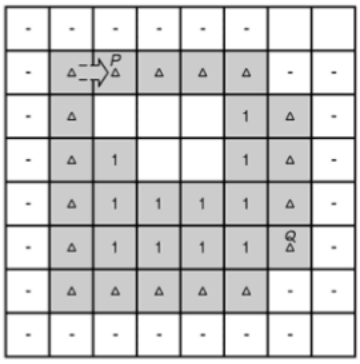


Figura 27: Tipos de píxeles 1:Píxeles negros sin etiquetar
△ : Píxeles negros etiquetados
-: Píxeles blancos marcados

Una vez marcados los píxeles blancos, además se asegura que cada contorno interno es trazado una sola vez. En la figura 28 se muestra como una vez trazado un contorno interno, el vecino debajo de R ya no es un píxel sin marcar, por lo tanto se evita hacer un trazo de contorno otra vez cuando R es encontrado por la línea de búsqueda. Es necesario trazar el contorno interno de un punto solo cuándo el píxel debajo del mismo es blanco y está sin marcar.

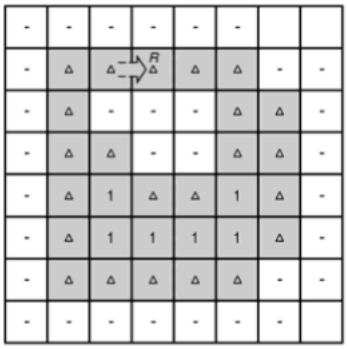


Figura 28: Los píxeles blancos son marcados cuando un contorno interno ha sido trazado

4.4.2. Trazado de Contorno

El objetivo del procedimiento del trazado de contorno es encontrar el contorno externo o interno de un determinado punto. Supongamos S, en el punto S se ejecuta el procedimiento de trazado, si en este proceso se identifica a S como un punto aislado, se termina el proceso de trazado de contorno, de otra manera, el proceso dará como resultado el punto de contorno que sigue a S, y asi sucesivamente.

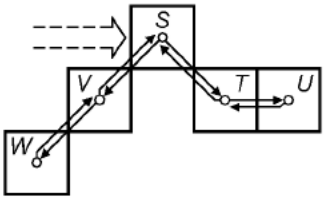


Figura 29: Ejemplo de Trazado de Contorno

4.4.3. Trazado

Para un punto de contorno P, el objetivo del trazado es encontrar ocho puntos vecinos a P. La posición de cada vecino es asignada e indexada como se muestra en la siguiente figura:

5	6	7
4	*P*	0
3	2	1

Figura 30: Asignación de vecinos en cvBlob

Cuando P no es un punto de inicio de un contorno independientemente de si el contorno es externo o interno, su posición de búsqueda inicial se establece por $d + 2(mod8)$, donde d es la posicion de el anterior punto de contorno.

Una vez determinada la posición inicial, se procede en sentido horario a localizar el primer píxel negro. Ese píxel sera el punto de contorno que sigue a P. Si no es encontrado ningún píxel en todo el círculo, P es identificado como un punto aislado.

4.5. Detector de blobs

Una vez entendido el proceso que realiza cvBlob, se aplicará la librería a nuestro proceso de tracking. Como se vio anteriormente, la imagen que será procesada por cvLabel debe de estar en

blanco y negro, lo cual es el resultado del proceso de segmentación a través de la función cvBGstatModel, la salida de este proceso será una imagen que estará etiquetada para saber los posibles blobs que se encuentran en la escena.

Ahora después de tener etiquetado el frame, cvBlob obtendrá los blobs empleando la técnica de componentes conectados y obtendrá, de la imagen etiquetada, cada uno de los blobs que están en la escena para su posterior procesamiento de seguimiento.

4.6. Tracking

Recordando lo que es el tracking, al obtener características ó identificación de características en el proceso de segmentación de una imagen, es normal que lo que se desea hacer a continuación es saber qué pasa con esos blobs que se tienen identificados, es decir, el movimiento asociado con esos objetos.

Al analizar las técnicas de tracking se logra distinguir algo muy importante, se deben asociar características o componentes entre un frame y otro para poder determinar que un objeto que aparece en el primer cuadro es el mismo objeto que avanzo, cierta distancia, en el siguiente frame.

Para este objetivo cvBlob emplea un tracking "sencillo"basado en [11], a un alto nivel, se puede explicar este proceso como el resultado del proceso de eliminación de componentes pequeños o aislados y el envolvimiento de componentes conectados en una caja.

Para cada sucesión de frames el objetvo es procesar las correspondencias de los píxeles envueltos y su respetivo tracking , esto se logra mediante la construcción de una matriz de distancias que muestra, valga la redundancia, la distancia entre cada uno de los objetos del primer plano y los que estan actualmente activos para realizar el tracking.

Se emplea una medida de las distancias de las cajas a partir de su centroide[12] y se calcula la distancia entre el centroide del objeto B a la ésquina del objeto A, como se muestra en la siguiente figura.

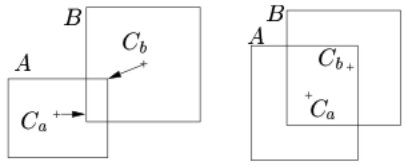


Figura 31: Medición de distancias entre las cajas

Posteriormente la matriz es binarizada, por medio de un umbral, resultando en una matriz

[12]Es el punto que define el centro geométrico de un objeto

correspondiente asociando un *track* con cada región del primer plano. Del análisis de la matriz se pueden identificar cuatro tipos de objetos:

- Objetos existentes. Representados por una correspondencia (filas identifican a un tracking existente, y columnas representan a las secciones del primer plano o blobs) de máximo un no cero en cada fila o columna, asociando cada tracking con cada blob.
- Nuevos objetos. Columnas con todos sus elementos en cero representan un nuevo objeto en la escena que no está asociado con algún track, y resulta en la creación de uno nuevo. Cabe mencionar que columnas con todos sus elementos en cero representan tracks que ya no son visibles, ya sea porque abandonaron la escena, o son regenerados por el proceso de substracción de fondo.
- Mezcla de objetos (oclusion). Dos o más tracks corresponderán a una región del primer plano, por ejemplo, una columna en la correspondiente matriz debe tener más de un no cero en sus elementos.
- Objetos divididos. Es el caso en el cúal en un grupo de personas, las personas se separan, un solo tracking corresponderá a múltiples blobs resultando en más de un no cero en la fila correspondiente de la matriz. Cuándo un solo track corresponde a más de una caja, todas las cajas se mezclarán y el proceso continuará. Si dos objetos seguidos se separan, las partes continuarán como uno solo, hasta que se separen lo suficiente para que cada una de las cajas sea considerado como un nuevo track.

En el código, lo referente a cvBlob se realiza en las siguientes líneas:

```
.
.
.
//Se realizará el etiquetado de la imágenes, y se guardarán los blobs correspondientes a cada cuadro de vídeo.
unsigned int result = cvLabel(morph, labelImg, blobs);
unsigned int result2 = cvLabel(morph2,labelImg2,blobs2);

//Se filtrarán los blobs encontrados, por área
cvFilterByArea(blobs,1000,10000);
cvFilterByArea(blobs2,1000,10000);

//Después del filtrado, se actualizará el proceso de tracking, asignando uno a cada blob en el frame
cvUpdateTracks(blobs,tracks,20,40);
cvUpdateTracks(blobs2,tracks2,20,40);

//Posteriormente se renderean los blobs en la imagen
cvUpdateTracks(blobs,tracks,20,40);
cvUpdateTracks(blobs2,tracks2,20,40);
```

5. Resultados

A continuación se muestran las pruebas realizadas con diferentes condiciones empleando nuestro algoritmo.

5.1. Tracking con 1 persona

Figura 32: Resultado del algoritmo de tracking con 1 persona

Las condiciones mostradas en la figura 32 incluyen filtraciones de luz, movimiento en las sombras proyectadas por los árboles, sombras del objetivo de tracking y como se observa la respuesta del algoritmo es buena.

Se observa que cuando la persona inicia su trayectoria, en la imagen superior izquierda, la respuesta en la detección de la misma es un poco retardada, pero conforme va avanzando en la secuencia (imagen superior derecha), la identificación es positiva y tiene un buen seguimiento incluso con el aumento de tamaño propiciado por el ángulo de la cámara, pero las cajas se adaptan al tamaño del objeto sin perder el ID de seguimiento.

Existe ruido provocado por el movimiento de los árboles, y aunque estos movimientos son detectados en el modelo de sustracción de fondo son anulados debido al filtro de tamaño empleado a través de cvBlob que excluye todos aquellos objetos que por ser demasiado pequeños, son ignorados como posibles prospectos para tracking.

En la parte final de la secuencia (imagen inferior derecha), existe un problema provocado por la cercania de la persona a la cámara, lo cual deriva en una partición del contorno y una doble identificación del mismo, un ID para la parte superior de la persona y otra para la parte inferior.

5.2. Tracking con 1 persona estática en la escena

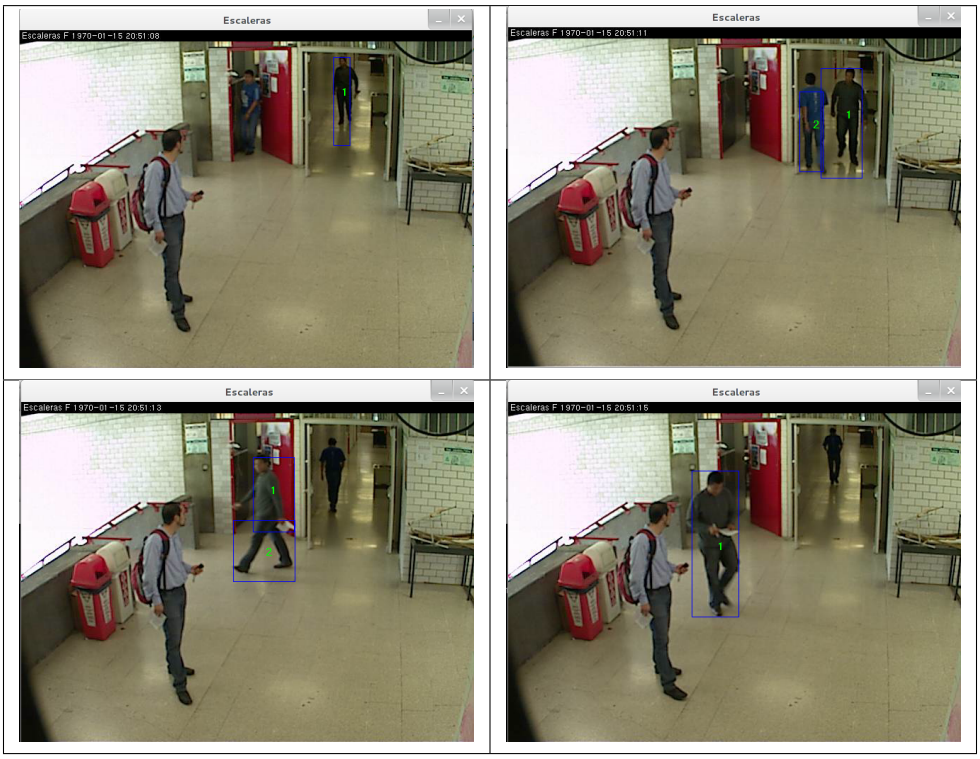


Figura 33: Resultado del algoritmo de tracking con 1 persona estática

La secuencia siguiente (figura 33) muestra 3 condiciones más en comparación al de la figura 32, existe una persona estática en la escena, se tienen dos luces, la natural proveniente del lado izquiero de las imágenes, y la artificial provocada por las lámparas en el pasillo (parte superior en cada imagen), lo cual en este entorno provoca una mayor presencia de sombras, y además existen 2 personas en movimiento.

El resultado es bastante bueno considerando que las personas en movimiento(imagen superior derecha) pasan una al lado de la otra y sin embargo la identificación es buena, aunque como en el caso anterior, existe un pequeño retardo en la respuesta de la identificación de la persona 2 y también se observa la partición del contorno (imagen inferior izquiera).

Cabe mencionar que si bien la persona estática no es tomada en cuenta, si en algún momento realiza un movimiento con los brazos, ese movimiento es marcado y se le da seguimiento por parte del algoritmo, pero sin perder el ID de los otros 2 ya detectados.

5.3. Tracking con oclusión

El siguiente caso de prueba es especial, ya que se probó el algoritmo en un entorno que es muy complicado de manejar en el tema de tracking, la oclusión.

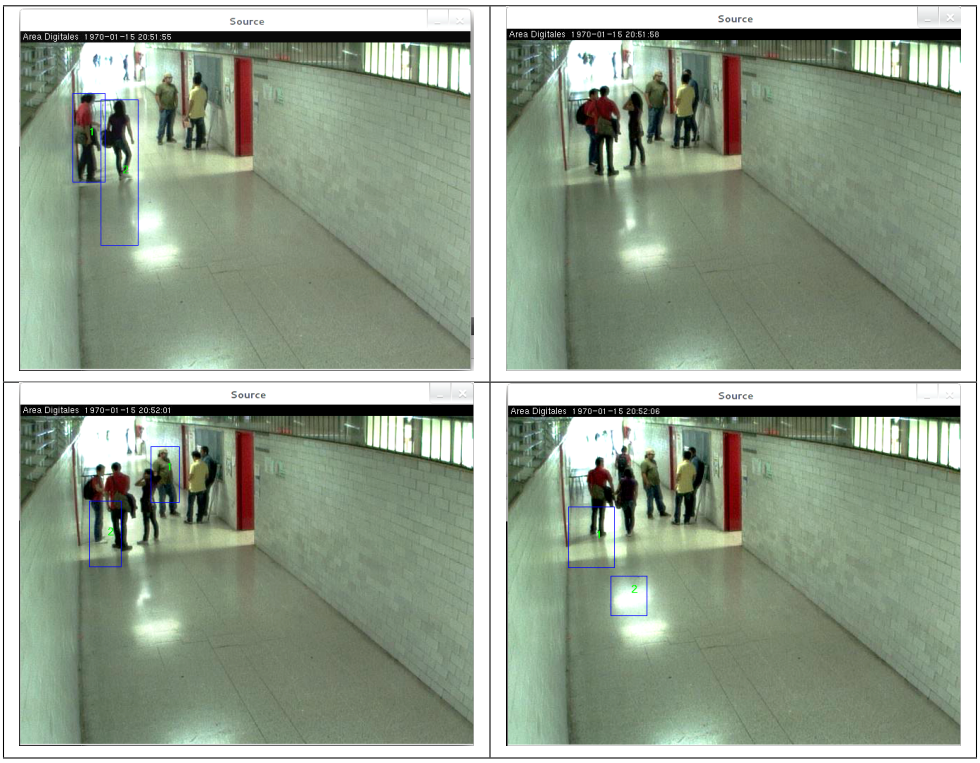


Figura 34: Resultado del algoritmo de tracking con oclusión

La figura 34 contiene problemas de luz mucho mayores, pues como se puede ver en el fondo de cada una de las imágenes, llega un punto donde las personas se pierden, también deriva en una mayor proyección de las sombras que ocasiona problemas en la detección de las mismas.

La imagen superior izquierda no muestra mayor problema en la aparición de las dos personas entrantes a la escena y su posterior identificación, salvo el problema con la gran sombra producida por la persona 2, pero se observa que el algoritmo no detecta las personas que están inmóviles. En el siguiente cuadro (imagen superior derecha), se obseva la oclusión en los dos grupos de personas presentes lo cual ocasiona una pérdida en el seguimiento provocando un reinicio en la búsqueda e identificación de los mismos.

La imagen inferior izquierda muestra el reinicio de la identificación, pero no de todas las personas, ocasionando (imagen inferior derecha) que al iniciar el movimiento de nueva cuenta se pierda el tracking e incluso se marca la identificación de una sombra, lo cual muestra una debilidad en el algoritmo para manejar esta situación.

5.4. Tracking en espacio abierto

Figura 35: Resultado del algoritmo de tracking en espacio abierto

La secuencia de la figura 35 fue obtenida con cámaras IP con una resolución de 640 x 480 píxeles, en la plaza roja de la Universidad Autónoma Metropolitana, a las 4 de la tarde, teniendo condiciones diferentes a las realizadas anteriormente, aqui se observa que es completamente al aire libre.

El gran problema a lidiar en este entorno fue el ruido, ya que tanto la calidad de la imagen como las condiciones ambientales provocan grandes puntos aislados y la poca definición de los contornos de la persona en movimiento.

El resultado es regular puesto que, aunque se cumple con la identificación de la persona en la escena, existe un retardo en la identificación y la continuidad de la misma, aunque el ID se mantiene a través de la secuencia.

5.5. Tracking multicámara

Figura 36: Resultado del algoritmo en tracking multicámara

La última prueba emplea los dos vídeos proporcionados del edificio G, uno es la continuación del otro, tomando en cuenta que la orientación no es la correcta, pues las personas salen del frame 1 por abajo de la escena y en lugar de aparecer por arriba del frame 2, también lo hacen por la

parte inferior del cuadro. Aplica el seguimiento en las dos secuencias de vídeo identificando a la persona a lo largo de los dos frames mediante el flujo de imágenes.

Si bien el entorno no cuenta con el control deseado, la respuesta del algoritmo en determinadas secciones del vídeo, funciona como se esperaba y sigue de manera sencilla a la persona que cambia de cuadro con el mismo identificador, aunque como se observa en la tercer secuencia, ocurre la partición en el segmentado, por consecuencia obtiene tres partes para una misma persona, pero después vuelve a establecerse el identificador como uno solo.

6. Conclusiones

El resultado en el proceso de pruebas, arroja las siguientes conclusiones:

- El modelo de segmentación muestra una buena respuesta, incluso con diferentes condiciones de luz, tanto en interiores como en exteriores.
- No se consideran personas que se encuentan estáticas.
- A pesar de tener condiciones de ruido, se eliminan condiciones no deseadas por medio del preprocesamiento y se anulan objetos que no son de interés por medio del filtro de blobs.
- El resultado de seguimiento es bueno e incluso se puede mostrar un ID para cada uno de los blobs.
- Cabe mencionar que el proceso se realiza haciendo una transformación a escala de grises, buscando reducir el proceso en términos de canales de color y definir mejor los objetos que se encuentran en la escena.
- Aunque se maneja ligeramente el caso de oclusión, se muestran errores en el momento que se separan las personas.
- En el proceso del tracking multicámara se observa que al pasar de un frame a otro, el tiempo de respuesta puede variar.
- En el caso específico de los grupos de personas, la respuesta puede variar, puesto que incluso con 2 personas en movimiento, la identificación es positiva, pero siempre y cuando no se entre en el caso de oclusión.

7. Trabajos Futuros

El presente trabajo se presta para realizar seguimientos y mejoras del mismo, las más importantes se listan a continuación:

- Mejoras al proceso de segmentación para evitar particiones del blob.
- Adaptación del proyecto dentro de un entorno de tiempo real.
- Adaptación del proyecto con cámaras IP.
- Mejoras al algoritmo de seguimiento en múltiples cámaras.
- Mejoras al algoritmo para el manejo de oclusión.
- Mejoras al algoritmo para el manejo de grupos de personas.

8. APENDICE

Como se mencionó en la proupesta de este proyecto, se utilizará OpenCV para realizar el procesamiento de los vídeos y se tendrá como base el entorno de desarrollo QtCreator para enlazarlo con la librería, a continuación se presenta el proceso de instalación de cada uno de ellos.

8.1. Compilación e Instalación de OpenCV

Basados en [2] se presenta esta guía de instalación esperando sea de ayuda para proyectos posteriores.

- Instalar los siguientes paquetes desde terminal
  apt-get install build-essential
  apt-get install cmake
  apt-get install pkg-config
  apt-get install libpng12-0 libpng12-dev libpng++-dev libpng3
  apt-get install libpnglite-dev libpngwriter0-dev libpngwriter0c2
  apt-get install zlib1g-dbg zlib1g zlib1g-dev
  apt-get install libjasper-dev libjasper-runtime libjasper1
  apt-get install pngtools libtiff4-dev libtiff4 libtiffxx0c2 libtiff-tools
  apt-get install libjpeg8 libjpeg8-dev libjpeg8-dbg libjpeg-prog
  apt-get install ffmpeg libavcodec-dev libavcodec52 libavformat52 libavformat-dev
  apt-get install libgstreamer0.10-0-dbg libgstreamer0.10-0 libgstreamer0.10-dev
  apt-get install libxine1-ffmpeg libxine-dev libxine1-bin
  apt-get install libunicap2 libunicap2-dev
  apt-get install libdc1394-22-dev libdc1394-22 libdc1394-utils
  apt-get install swig
  apt-get install libv4l-0 libv4l-dev
  apt-get install python-numpy
  apt-get install libjpeg-progs libjpeg-dev
  apt-get install libgstreamer-plugins-base0.10-dev

- Obtener la última versión estable de OpenCV de la dirección `http://sourceforge.net/projects/opencvlibrary`

- Existen varias formas de realizar la compilación, por conveniencia y debido al entorno gráfico que presenta necesitaremos instalar los siguientes paquetes:
  apt-get install cmake cmake-gui

- Una vez instalado, abrimos cmake y seleccionamos la fuente, en este caso el archivo comprimido de OpenCV, y el destino,donde irán los archivos extraídos, si lo desean en la carpeta /opt/ o en el /home/, dependiendo los permisos.
  Aquí podremos seleccionar entre las opciones de compilación, las librerías que deseamos que

utilice, las rutas donde se extraerán los archivos entre otras cosas. Configuramos y generamos los archivos con los botones correspondientes.

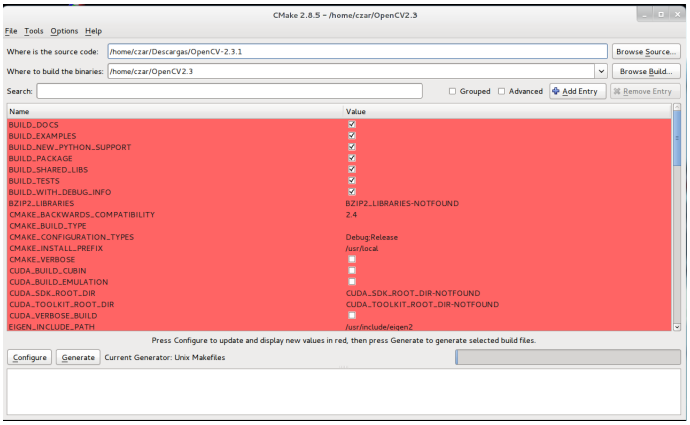


Figura 37: Ventana de Cmake

- Después desde el directorio en dónde se encuentra OpenCV se abre una terminal y se ejecuta el comando *make* y si no existen errores se procede con *make install*.
- Posteriormente se edita el archivo */etc/ld.so.conf* agregando la ruta */usr/local/lib/* para que la librería se encontrada.
- Por último ejecutamos *ldconfig -v* para cargar la nueva configuración.

8.2. Instalación de QtCreator y configuración con OpenCV

Una vez instalado OpenCV, se procede a instalar QtCreator[3], que se encuentra directamente en los repositorios, si se desea trabajar con la última versión se descarga de la siguiente página `http://qt.nokia.com/downloads/`.

Ya dentro del IDE se creará un proyecto de consola y se crearán automáticamente 2 archivos, en el archivo con extensión .pro se agregarán las rutas de las librerías de OpenCV quedando de la siguiente forma.

8.3. Compilación e instalación de cvBlob

En [4] se describe como realizar este proceso, aqui se mencionan las operaciones brevemente.

- Descomprimir la fuente de cvBlob descargado de [4].
- Dentro del directorio realizar un *make*.
- Posteriormente realizar un *make install*

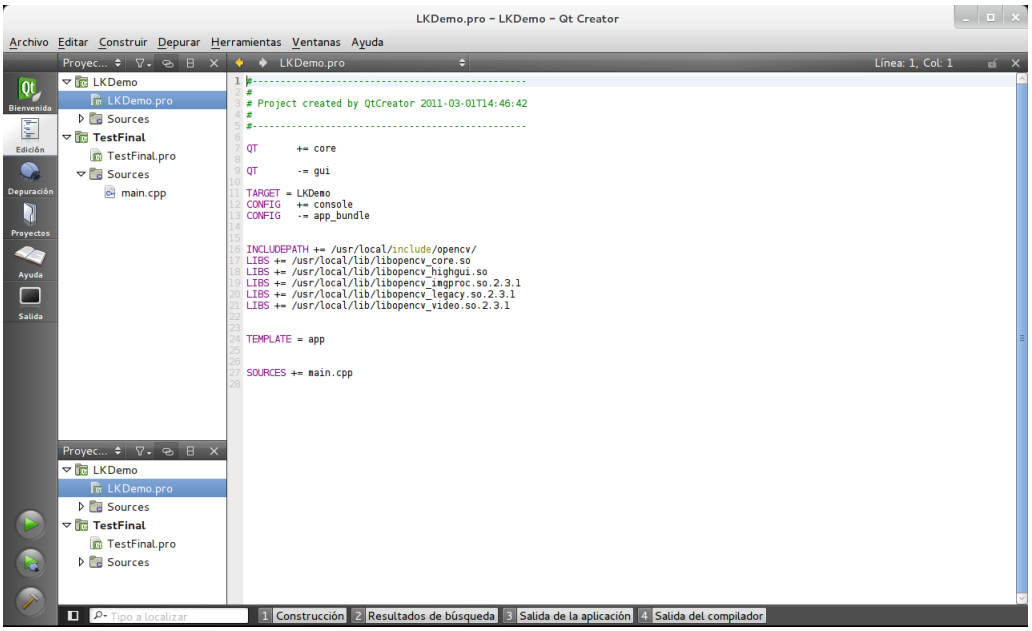


Figura 38: OpenCV en QtCreator

- Referenciar la librería en QtCreator, como se hizo con OpenCV.

9. Bibliografía

Referencias

[1] CVLAB-Computer Vision Laboratory, *"Tracking People using Multiple Cameras".* `http://cvlab.epfl.ch/research/body/surv/`. Consultada en octubre del 2011.

[2] OpenCVWiki `http://opencv.willowgarage.com/wiki/InstallGuide%20:%20Debian`.Consultada en Noviembre del 2010.

[3] QT Creator Integrated Development Enviroment. `http://qt.nokia.com/downloads`. Consultada en noviembre del 2010.

[4] cvBlob `http://code.google.com/p/cvblob/`. Consultada en marzo del 2011.

[5] Mplayer-Mencoder `http://www.mplayerhq.hu/DOCS/HTML/es/index.html`.Consultada en noviembre del 2010.

[6] Liyuan Li, Weimin Huang, Irene Y.H. Gu, Qi Tian, *"Foreground Object Detection from Videos Containing Complex Background"* `http://opencv.willowgarage.com/wiki/VideoSurveillance`.Consultada en noviembre del 2010.

[7] Connected-component labeling `http://en.wikipedia.org/wiki/Connected_Component_Labeling`. Consultada en marzo 2011

[8] Fu Chang, Chun-Jen Chen and Chi-Jen Lu, *"A linear-time component-labeling algorithm using contour tracing technique".*,Institute of Information Science,Academia Sinica, Taipei Taiwan.

[9] XviD `http://www.xvid.org/FAQ.14.0.html`

[10] Yao Jian, Odobez Jean-Marc, *"Multi-Camera Multi-Person 3D Space Tracking with MCMC in Surveillance Scenarios"*, 16th European Signal processing Conference, EUSIPCO 2008.

[11] A. Senior, A. Hampapur, Y-L Tian, L. Brown, S. Pankanti and R. Bolle, *"Appeareance Models for Occlusion Handling"*, IBM T.J. Watson Research Center, Yorktown Heights,NY.

[12] Q.Cai and J.K. Aggarwal, *"Tracking Motion Using Multiple Cameras"*, Computer and Vision Research Center, The University of Texas at Austin.

[13] Bradski Gary, Kaehler Adrian, *"Learning OpenCV"*, 1ra Ed. O'Reilly, 2008. ISBN 978-0-596-51613-0.

[14] Rafael C. González, Richard E. Woods, *"Tratamiento Digital de Imágenes"*, Addison-Wesley Iberoamericana,1996. ISBN 0-201-62576-8

[15] Gonzalo Pajares Martinsanz, Jesús M. de la Cruz García, *"Visión por computador. Imágenes Digitales y Aplicaciones"*, 2^a Ed. AlfaOmega, 2008. ISBN 978-84-7897-831-1