Análisis de temporalidad de imágenes con técnicas de detección, tracking y proyección de posición de un objeto.

Author
Leonardo M. Bustamante[1], Marcos Maciel[1], Daniela López De Luise[2]
[1]CAETI, Buenos Aires, Argentina
[2]CI2S Labs, Buenos Aires, Argentina

Resumen

Es importante para la toma de decisiones basadas en hechos pasado con efecto en el presente y consecuencias a futuro contar con información anticipada que permita ejecutar acciones. Este trabajo presenta un modelo orientado a anticipar un evento por medio de la detección de objetos de interés, el tracking de estos y la proyección de su posición a futuro con respecto a una cámara observadora, esto se logra mediante el procesamiento de imágenes de baja resolución obtenidas por un teléfono móvil, incluso de baja gama. Es de destacar que en el presente trabajo, la detección no está limitada a objetos previamente configurados y/o entrenados dentro del modelo. Cualquier tipo de objeto en movimiento está incluido. Las estadísticas realizadas demuestran un porcentaje de detección de objetos y proyección aceptable para el hardware de bajo costo empleado.

Palabras clave: Imágenes, Procesamiento de Imágenes, Reconocimiento de Objetos, Tracking, Proyección.

I. Introduction

La detección del movimiento de objetos tiene innumerable aplicaciones, cada una con sus ventajas y desventajas. Existen aplicaciones orientadas a resolver un problema en particular que no pueden ser usadas en otras áreas. Es posible identificar un objeto por su color y seguirlo por distintas posiciones para determinar su movimiento [01], otra estrategia muy utilizada es reconocer figuras a través de la segmentación de imágenes digitales, agrupando los puntos (pixeles) que pertenecen a los bordes usando la transformación de Hough [02].

Otras estrategias resultan específicas pero interesantes, como la detección de rostros propuesta por [03] a través de clasificadores en cascada enfocada en el análisis de las regiones con mayor probabilidad de cubrir los objetos de interés, en este trabajo rostros. Como en otros casos, el problema es que la detección está limitada por la iluminación, oclusión en ojos y boca.

Con el fin de superar los problemas en la variación de iluminación la propuesta de [04] identifica peatones, para ello calcula el histograma de regiones permitiendo desde los contornos determinar la forma de los objetos. Finalmente clasifica los bloques descriptores, llamados Histogram of OrientedGradients (HOG), usando la máquina de soporte vectorial. [05] Esta aproximación, sin embargo, genera ruido en el cambio en tamaño de los objetos.

En [06] se desarrolla un método que encuentra objetos en una imagen con apariencias muy variadas utilizando la búsqueda de ventana deslizante y representando en cada región, estructuras jerárquicas de partes o filtros raíz. Se tienen en cuenta las diferentes secciones que componen un objeto y pueden identificar a este objeto en diferentes posturas. Su problema es el requerimiento computacional pues cuantas más secciones se usen para componer un objeto, será necesario más tiempo para procesar el algoritmo.

La búsqueda de regiones en ventana deslizante tiene como desventaja el procesamiento de muchas regiones, además del uso de clasificadores que suman tiempo al procesamiento de cada imagen. La alternativa es trabajar sobre regiones candidatas o subregiones porque dentro de estas la probabilidad de encontrar estos objetos es mayor, además que se utilizan conjuntos con menos regiones candidatas y clasificadores más complejos, todo esto en conjunto tiene mejor rendimiento y performance.

Una implementación de esta técnica es propuesta en [07], permite determinar si un objeto (de

diversas clases) se encuentra en una región previamente identificada en la ventana deslizante. Para cada región se examinan factores como color, bordes y similaridad en textura/color y de acuerdo a estos parámetros se estima un puntaje para la ventaja deslizante.

En la actualidad se hace uso del aprendizaje profundo (deeplearning) como estrategia para el análisis de imágenes aprovechando las capacidades de hardware existentes. Éstas hacen uso de las Redes Neuronales Convolucionales con muy buena performance [08]. A manera de ejemplo se puede mencionar a YOLO (YouOnly Look Once) [09]. La detección de objetos es tratada como un problema de regresión simple, donde de cada dato de entrada (imagen) se aprende cuando se encuentra una clase en una zona, la que se representa como coordenadas dentro de una región.

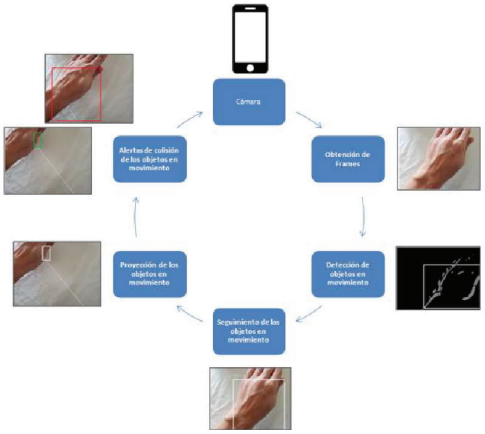
El modelo propuesto en este trabajo está enfocado en detectar un o varios objetos de interés, hacer un tracking, una proyección de localización y en base a esta información generar eventos de alerta que pueden ser gestionados de diversas manera por suscriptores de interés. Otra particularidad propuesta es la utilización de este modelo en teléfonos móviles incluso de baja gama y de fácil acceso.
Por último y no menos importante la detección de objetos no incluye su reconocimiento por tipo o forma ya que no es de interés para este modelo.
Este paper está organizado de la siguiente forma: descripción del modelo propuesto en la sección II, en la sección III se encuentra el desarrollo del testing al modelo y por último se hallan conclusiones y trabajos futuros en sección IV.

II. Modelo Propuesto

El modelo propuesto en este documento utiliza el exitoso framework OpenCV © [10] que es una versión libre desarrollada por Intel desde el año 2000, y que hoy se utiliza en sistemas de seguridad y control de procesos. El prototipo se halla codificado en Java © para posibilitar su uso en dispositivos móviles. Existen otros frameworks que realizan esta tarea, pero se decide desarrollar un nuevo modelo con la posibilidad de optimizar los procesos según el rendimiento en los dispositivos móviles.

En base a una investigación basada en un método heurístico, se identifican 5 procesos individuales que secuenciados logran una detección y seguimiento razonable, permitiendo predecir con precisión suficiente su trayectoria. Los procesos son los presentados en la Fig. 1.

Fig. 1. Esquema global de procesamiento de imágenes



En las subsecciones que siguen se describe cada una de estas etapas.

II.A. Obtención de Frames

Este proceso se encarga de obtener un frame de la cámara del teléfono móvil. Dependiendo del modelo del teléfono, la cámara puede brindar entre 10 y 30 frames/segundo. Incluso, se pueden registrar casos en que las imágenes van desde los 5 Mp con un resolución de 480x800 píxeles, a otras de 12 Mp con una resolución de 4032 x 1960 pixeles.

OpenCV © brinda un "bridge" para administrar la cámara de un dispositivo móvil y configurar su resolución [11]. Para el prototipo que implementa el modelo de este proyecto, se trabaja con una resolución a 320 x 240 pixeles y se convierte los frames en formato RGB a un formato de escala de grises con el objeto de acelerar el análisis de la imagen [12]. Para convertir a escala de grises se utiliza el framework OpenCV © que implementa la ecuación ec.1.

$$\text{RGB[A] to Gray:} \quad Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \qquad \textbf{(ec.1)}$$

Por cada canal alfa "A" de la imagen en formato RGB se suman el resultado de multiplicar cada componente "R", "G" y "B" por su correspondiente constante. Se obtiene así su homólogo en escala de grises.

En el caso de los testeos realizados, se mide a una tasa de 30 frames/segundo, dando lugar a un objeto tipo MAT que brinda el framework OpenCV © y que se almacena en memoria.

A los fines estadísticos, se almacenan estos frames en memoria como objetos tipo MAT con archivos de imágenes formato .png, con la misma resolución (320 x 240 píxeles) generando archivos entre 70 Kb y 85 Kb.

II.B. Detección de Objetos en Movimiento

Este proceso se encarga de comparar la secuencia de imágenes obtenida en la sección anterior, para luego detectar objetos en movimiento.Esto se logra aplicando el filtro "Gaussian Mixture Model" basado en la mezcla de la imagen de fondo y una imagen de la secuencia [13] a través de la función "BackgroundSubtractorMOG2" que brinda el framework OpenCV ©. Los valores usados en esta fucion son todos por default.La Fig. 2, presenta un ejemplo pequeño con sólo cuatro figuras.

Fig. 2. Secuencia de cuatro imágenes resultante de la aplicaciòn del filtro Gaussian Mixture Model.

Como se puede apreciar, la secuencia presenta el desplazamiento físico en el tiempo de una mano izquierda de un lado a otro. En la primera imagen aún el objeto no llega a plano de trabajo, y en los siguientes, se aprecia el resultado de la resta entre la imagen de la izquierda, y la actual. En el caso de haber fondos no lisos, éstos serán eliminados, incluso con parte de la imagen de interés, cuando la misma se confunda inicialmente con el fondo.

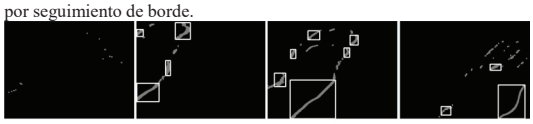
El siguiente paso es tomar la imagen del paso anterior y analizar y detectar las superficies grises. Esto se logra haciendo un análisis topológico de la imagen por seguimiento de bordes a través de la función "findcontours" que brinda del framework OpenCV © que genera una secuencia de puntos que rodean la superficie gris [14]. La Fig. 3, presenta un ejemplo del resultado del análisis.

Fig. 3. Secuencia de cuatro imágenes resultante del análisis topológico por seguimiento de borde.

Se puede apreciar que las superficies grises son rodeadas por una secuencia de puntos blancos formando un contorno. Esta secuencia de puntos se almacenan en un array de objetos del tipo MAT OF POINT brindado por el framework OpenCV © en memoria. Cada Objeto del tipo MAT OF POINT posee la coordenada X e Y del punto respecto del centro del eje de coordenada de la imagen que es el ángulo superior izquierdo. El resultado de este paso son los contornos de las partes detectadas del objeto en movimiento.

Una secuencia de puntos que rodea una superficie gris en una imagen representa una cantidad significativa de valores que es un costo mantenerlos en memoria y poco performante para el análisis. El siguiente paso encuentra un recuadro que contenga la secuencia de puntos que forman el contorno. Se recorre el array de objetos del tipo MAT OF POINT de cada contorno detectado y se busca el valor máximos y mínimo de las coordenadas X e Y de cada punto. Como resultado de este proceso se obtiene el ángulo superior izquierdo y el ángulo inferior derecho que forma el rectángulo que contiene todo los puntos que rodean a un contorno. Esto se logra, utilizando la función "BoundingRect" que brinda el framework OpenCV © [15]. La Fig. 4, presenta un ejemplo del resultado del análisis.

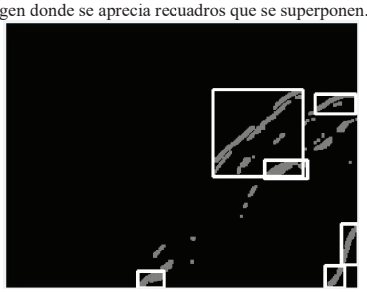
Fig. 4. Secuencia de cuatro imágenes resultante del análisis topológico por seguimiento de borde.



Se puede apreciar en cada imagen de la secuencia, los contornos detectados del objeto en movimiento, es rodeado por un rectángulo. El resultado de este proceso es que reemplaza el array de puntos que rodea un contorno, por dos coordenadas que forma un rectángulo que los contienen.

Como consecuencia del paso anterior, se obtienen recuadros que se superponen entre ellos. La Fig 5, brinda un ejemplo del problema mencionado.

Fig. 5. Imagen donde se aprecia recuadros que se superponen.



Para resolver el problema mencionado, se aplican un ratio llamado IntersectionoverUnion (IoU) que calcula la proporción que comparten entre los recuadros [16]. En la

ecuación ec.2, se presenta la ecuación empleada por el filtro durante el proceso IoU.

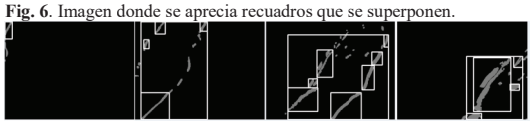
$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad \text{(ec. 2)}$$

Area of Overlap es área intersección entre los recuadros comparados, mientras que Area of Union es la unión de las áreas de los recuadros comparados menos su intersección.IoU es una medida métrica sin unidades puesto que tan sólo establece una tasa de proporción.

Para el modelo, se descartan recuadros con un área menor a 100 pixeles, y se considera el mismo recuadro cuando IoU es mayor a cero. Estos dos pasos logran limpiar razonablemente los errores por la fragmentación obtenida de la detección de contornos.

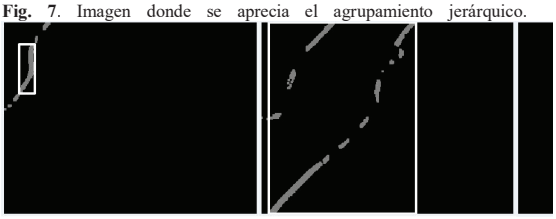
Finalmente, se implementa un algoritmo de agrupamiento jerárquico que agrupa los recuadros que están cerca de menos de 100 pixeles respecto de sus centros geométricos [17]. La Fig. 6, presenta un ejemplo de agrupamiento de recuadros cercanos.

Fig. 6. Imagen donde se aprecia recuadros que se superponen.



Se puede apreciar cómo los recuadros pequeños cercanos entre sí son agrupados por un recuadro más grande que los contiene. En la Fig. 7 se muestra un ejemplo del agrupamiento obtenido.

Fig. 7. Imagen donde se aprecia el agrupamiento jerárquico.



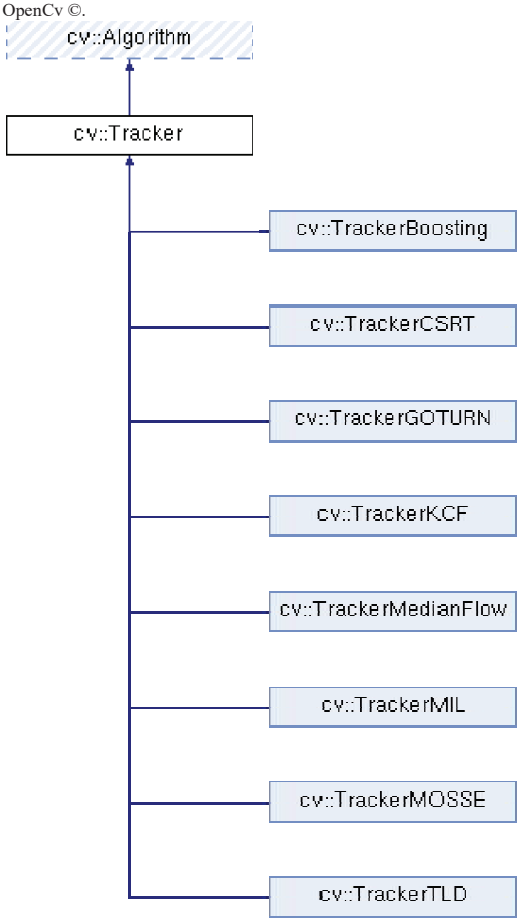
Las imágenes muestran el resultado final del proceso de detección, que es un recuadro que selecciona el objeto en movimiento a través de una secuencia de imágenes.

II.C. Tracking de las Detecciones

Con el objetivo de lograr predecir la posición de un objeto en movimiento, un paso importante es lograr identificar el mismo objeto en movimiento a través de la secuencia de imágenes. Este proceso se lo denomina "Tracking" o seguimiento, y permitirá analizar todas las posiciones del objeto y predecir su localización en un tiempo futuro. El modelo logra ésto utilizando el filtro KernelizedCorrelationFilters (KCF) que permite estimar las coordenadas de un objeto en la siguiente imagen de la secuencia.

Para llevar adelante este proceso, se utiliza el filtro de tracking que proporciona el framework de OpenCV© [18]. La Fig. 8, muestra la jerarquía de clases y las distintas implementaciones.

Fig. 8. Representación de las distintas jerarquías de Tracking que brinda OpenCv ©.



Con el objetivo de seleccionar un algoritmo de tracking, se realizan pruebas con cada uno de ellos con un set de 900 imágenes. Por cada imagen se detectan los objetos en movimiento según el método que se explica en la sección II.B. Al mismo tiempo, se alimenta el algoritmo de tracking en prueba con cada detección para que analice y realice la predicción de la posición del objeto en la siguiente imagen. Finalmente, se define un indicador denominado "Proyecciónes Positiva" que contabiliza cuando la detección y la predicción son "Similares". Se define como "Similares" cuando el ratio IoU definida en la ec.2 entre la detección y la proyección es mayor o igual a 0.7. Frente a los resultados, el algoritmo que muestra mejores aciertos es TrackeKCF (KernelizedCorrelationFilters) [19]. La Fig. 9 se muestra los resultados del test.

Fig. 9. Resultados de la prueba de los algoritmos de tracker.

Algoritmos	Proyecciones positivas
TrackerBoosting	64%
TrackerCSRT	17%
TrackerGOTURN	47%
TrackerKCF	85 %
TrackerMediaFlow	12%
TrackerMIL	8%
TrackerMOSSE	73%
TrackerTLD	67%

El modelo realiza dos acciones: Agregar y Actualizar. Cada imagen acompañada por su lista de detecciones obtenidas del proceso anterior, el modelo analiza si cada detección pertenece a un objeto anterior o no. En caso de que no, se utiliza la acción "Agregar" para comenzar hacer el tracking de un nuevo objeto detectado. En caso de que si, el modelo utiliza la acción "Actualizar" para agregar una nueva posición del objeto en seguimiento.

Para lograr identificar si un nuevo objeto detectado es un objeto detectado anteriormente, el modelo utiliza el algoritmo IoU (IntersectionoverUnion) que compara cada nuevo objeto detectado con cada posible proyección futura de un objeto, brindado por el algoritmo "TrackerKCF". Si la proyección y la nueva detección comparten el mismo lugar en la imagen, el modelo supone que es el mismo objeto.

El modelo lleva adelante un proceso para depurar los objetos en seguimiento, lleva un contador de cuántas imágenes se analizaron y cuáles no logran hacer una actualización de su nueva posición. En base a las experiencias realizadas, se determina un valor umbral promedio. Si este valor es superado indica que se perdió el rastro del objeto en seguimiento y el modelo descarta su rastreo.

Este modelo presenta errores basados en el movimiento de la cámara cuando toma la nueva imagen y desfasa el eje de coordenadas. Para controlar esto, se aprovecha el proceso de depuración explicado anteriormente. Se aumeta el valor umbral que permite que se pierda el rastro de un objeto por más cantidad de frames, dando la oportunidad de volver a encontrarlo.

En la siguiente secuencia de imágenes, se muestran las distintas detecciones identificadas con un ID. Al correr la secuencia, es posible notar que mantiene el mismo valor a través del tiempo por la lógica de "Tracking" o seguimiento. Aun cuando se pierde una secuencia, si el objeto vuelve a plano de trabajo, logra recuperar el ID original. La Fig. 10, muestra un ejemplo de la identificación en una secuencia de cuatro imágenes.

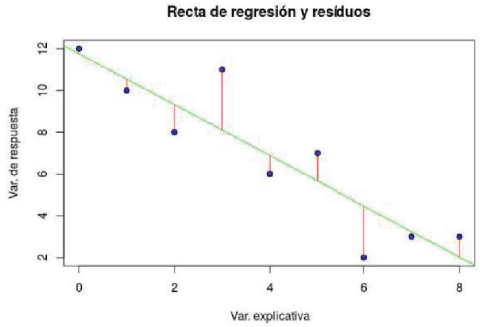
Fig. 10. Identificación de objeto detectado en el tiempo.

En la primera imagen se detecta un objeto en movimiento. El tracking le asigna el siguiente ID disponible, en el tiempo. Por el seguimiento y uso del filtro "TrackerKCF", se determina que es el mismo objeto y por lo tanto mantiene el ID anterior. Asumiendo que la tercera imagen es la vuelta a plano de trabajo en la última se asigna el ID original.

II.D. Proyección del Movimiento

El objetivo de este proceso es analizar todos los movimientos de un objeto detectado en las distintas imágenes analizadas y predecir su trayectoria. Para realizar este análisis, el modelo utiliza una función de "Regresión Lineal Simple", la que toma cada posición del objeto y proyecta en línea recta su posición a futuro [20]. La fig. 11, muestra el análisis resultante por este modelo de predicción del comportamiento de un objeto en movimiento en los distintos momentos en que es observado.

Fig. 11. Gráfico de la recta de regresión y residuos.



El gráfico muestra el comportamiento de cada posición del objeto, se calculan los coeficientes a (ordenada al origen) y b (pendiente de inclinación de la recta) de acuerdo a ecuación ec.3 que predice su posición en el eje "Y" según su posición en eje "X".

$$Y_i = (a + bX_i) + \epsilon_i$$

(ec.3)

Las líneas de color rojo de la Fig. 10 se denomina error y está dado por la ecuación **ec.4**

$$Y - \hat{Y} = \text{error} \quad \textbf{(ec.4)}$$

El modelo contempla un margen de error adicional dados por el desplazamiento de la cámara mencionado anteriormente y los cambios de velocidad del objeto como así también el cambio de velocidad del observador. La Fig. 12, se observa el margen de error adicional.

Fig. 12.Las dos imágenes representan el desplazamiento de una mano izquierda, su detección y proyección.

El desplazamiento de una mano visto con una cámara fija tiene asociado un margen de error propio del movimiento del objeto, pero es considerado por este modelo un margen de error adicional si la cámara deja de ser fija y además tiene movimientos oscilantes y aleatorios.

II.E.Alertas

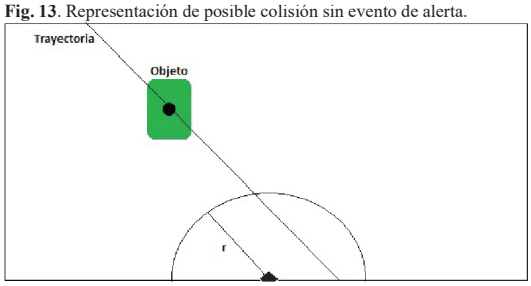
Para este proyecto, se define un alerta. Este alerta es cualquier tipo de comunicación o aviso desde el software que está ejecutando las distintas rutinas anteriormente descritas. Con este aviso se pretende que otro software pueda hacer uso de esta información para tomar alguna decisión. Por definición alertar indica si el objeto detectado y en seguimiento se encuentra en una trayectoria de posible colisión con el observador (cámara que toma las imágenes). Para lograr ese objetivo, se implementa un radio de seguridad de r (radio) medido en pixeles desde la posición central e inferior de la imagen o posición del observador.

Las condiciones que este modelo impone para generar un evento de alerta son:

- r=100 pixeles, i= valor entero en un rango de 1-100 expresado en porcentaje.
- El objeto conserva en dirección descendente o direccionada al observador.
- La trayectoria del objeto cruza la zona de seguridad definida por r.
- La distancia de la última posición del objeto al centro de la zona de seguridad es menor o igual a m=(r*i)/100 pixeles.

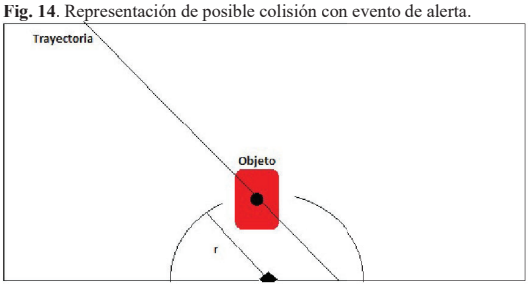
Estas condiciones se emplean para generar reglas de un sistema experto que decida cuándo desencadenar un evento de alerta o no. Las Fig. 13 y 14 muestran las distintas alternativas.

Fig. 13. Representación de posible colisión sin evento de alerta.



El observador (cámara del teléfono móvil) se encuentra en la base de la imagen. El objeto detectado se dirige hacia el área de interés marcado con un círculo. La trayectoria indica que la colisión es posible. La función de regresión lineal simple determinada por la recta muestra que el objeto no va directo al punto de interés. En este caso el valor asignado a la variable i determina que ningún alerta debe dispararse.

Fig. 14. Representación de posible colisión con evento de alerta.



El observador (cámara del teléfono móvil) se encuentra en la base de la imagen. El objeto detectado se dirige hacia el área de interés marcado con un círculo. La trayectoria indica que la colisión es posible. La función de regresión lineal simple determinada por la recta muestra que el objeto no va directo al punto de interés. En este caso el valor asignado a la variable i determina que el evento de alerta debe ser originado. Estos casos son específicamente administrados por las necesidades del implementador de este modelo y se codifican adecuadamente en el sistema experto para que éste tome las decisiones del caso de manera automática.

III. Resultados Obtenidos

Se han realizado pruebas sobre 2.399 imágenes o frames extraídos de un video filmado en ambiente exterior. Cada imagen tiene una resolución de 4.032 x 1.960 pixeles. Al aplicar el modelo propuesto se reduce el tamaño de cada frame a 320 x 240 pixeles y se logra procesarcon una tasa de 30 frames/segundo. Los escenarios incluidos en la muestra utilizada para hacer el testing son:

- Veredas.
- Calles.
- Tránsito.
- Personas caminando.
- La iluminación de días soleados, nublado y combinado.

Para estas pruebas se definió un proceso de "Alerta" en el modelo propuesto con la siguiente configuración:

- r=100 pixeles, i=100 pixeles.
- El objeto detectado está en dirección descendente (desde la perspectiva de la cámara).
- La trayectoria del objeto cruza la zona de seguridad (r de 100 pixeles).
- La distancia de la última posición del objeto al centro de la zona de seguridad es menor o igual a m=(r*i)/100 pixeles.

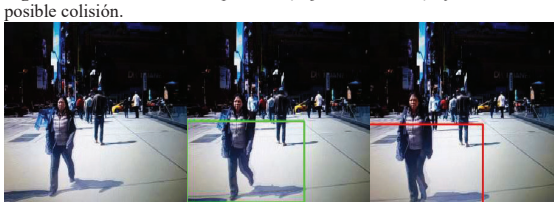
Sobre este conjunto de prueba, en primer lugar se identifican visualmente los objetos en movimiento y determinan los eventos que deben generar un alerta. En la siguiente etapa se ejecuta el modelo propuesto. La Fig. 15, muestra los resultados en forma de cuadro.

Fig. 15. Cuadro comparativo: eventos de alerta detectados manualmente y los desencadenados por el modelo propuesto.

	Detecciones Correctas	Deteccion es Falsas
Análisis Manual	851	
Análisis Algoritmo	671	127
	78,85 %	

Dando por ciertos todos los eventos de alerta manual (851 casos), los eventos de alerta desencadenados por el modelo propuesto (671 casos) se pueden estimar en una tasa de efectividad del 78.85%. En la Fig. 16, se puede observar un caso de detección (recuadro verde), tracking, proyección y evento de alerta (recuadro rojo).

Fig. 16. Detección de una persona (objeto de interés), y alerta de posible colisión.



En la escena 1 se determina una persona de interés.En la segunda por comparación se registra el movimiento y la dirección al observador. Por último, el modelo desencadena un evento de alerta mediante un recuadro rojo.

Frente a estas pruebas, el porcentaje de efectividad del modelo presente es suficiente para las implementaciones en que deriva este trabajo de investigación. Sin embargo, la tasa de efectividad debe ser mejorada en casos que la seguridad sea un factor clave.

Por otro lado, las pruebas del modelo detectan falsos positivos (ver la Fig. 17). Sobre el set de imágenes de 2.399 imagenes, 1.548 no contienen detecciones. El modelo detecta 127 (8,2%) detecciones falsas. Una de las causas identificadas, es la iluminación que genera la sobra de una persona sobre el piso, ésta es detectada por el algoritmo.

Fig. 17. Detección de falsos positivos.

IV. Conclusión y trabajos futuros

En este primer trabajo se presenta un conjunto de funciones de software ejecutados en un teléfono móvil que implementa un modelo de inferencia lineal sobre imágenes desplazadas. El principal aporte de este modelo, es detectar cualquier tipo de objeto en movimiento. Hacer un rastreo (tracking) específicamente determinado en el momento y proyecta a futuro su localización en el plano de trabajo. Con toda esta información se puede desencadenar un evento de alerta o predicción de colisión para que quien implementa esta solución pueda ejecutar su propia rutina de toma de decisión o acción. Como trabajo futuro se buscará disminuir el ruido generado por la iluminación del ambiente exterior, ejecutar el modelo en ambiente exterior extremo y con diferentes tamaños de objetos de interés e inclusive un mix de ellos. También resta investigar el uso de distintos dispositivos móviles (alta y media gama) para acelerar la velocidad de procesamiento, la cantidad de frames por segundo que resulten críticos en cada caso y la sensibilidad a la resolución de las imágenes (en píxeles).

Referencias

[01] Barnard, K., Cardei, V., &Funt, B. (2002). A comparison of computational color constancyalgorithms - Part I: Methodology and experimentswithsynthesized data. IEEE TransactionsonImageProcessing, 11(9), 972–984. https://doi.org/10.1109/TIP.2002.802531

[02] Canul-arceo, L., López-martínez, J., & Narváez-díaz, L. (2015). Algoritmo rápido de la transformada de Hough para detección de líneas rectas en una imagen Fastalgorithm of the Hough transformforstraight line detection in animage. Programación Matemática Y Software, 7(2), 8–13.

[03] Viola, P., & Jones, M. (2014). Robust Real-Time FaceDetectionRobust Real-time FaceDetection, (May 2004), 2–3. https://doi.org/10.1023/B

[04] Dalal, N., &Triggs, B. (n.d.). Histograms of OrientedGradientsfor Human Detection.

[05] Boser, B. E., Laboratories, T. B., Guyon, I. M., Laboratories, T. B., &Vapnik, V. N. (n.d.). A Training AlgorithmforOptimalMarginClassiers.

[06] Felzenszwalb, P. F., Girshick, R. B., Mcallester, D., &Ramanan, D. (n.d.). ObjectDetectionwithDiscriminativelyTrainedPartBasedModels, 1–20.

[07] Alexe, B., Deselaers, T., & Ferrari, V. (n.d.). Whatisanobject ?

[08] He, K., &Sun, J. (n.d.). Deep Residual LearningforImageRecognition, 1–9.

[09] Redmon, J., Girshick, R., Farhadi, A., &Dataset, A. (n.d.). YouOnly Look Once : Unified , Real-Time ObjectDetection.

[10] OpenCV: https://opencv.org/, https://github.com/opencv/opencv, (2017)

[11] OpenCV, CameraBridgeViewBase, https://docs.opencv.org/2.4/doc/tutorials/introduction/android_binary_package/dev_with_OCV_on_Android.html?highlight=camerabridgeviewbase.

[12]Chen, Peijian, MovingObjectDetectionBasedonBackgroundExtraction, IEEE, (2009).

[13] OpenCV, BackgroundSubtractorMOG2, https://docs.opencv.org/3.4/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html.

[14] OpenCV, FindContours, https://docs.opencv.org/3.4.0/df/d0d/tutorial_find_contours.html.

[15] OpenCV, BoundingRect, https://docs.opencv.org/3.4.0/dd/d49/tutorial_py_contour_features.html.

[16] Wikipedia, IntersectionoverUnion, https://en.wikipedia.org/wiki/Jaccard_index.

[17] Wikipedia, Agrupamiento jerárquico, https://es.wikipedia.org/wiki/Agrupamiento_jer%C3%A1rquico.

[18] OpenCV, Tracking, https://docs.opencv.org/3.4.1/d0/d0a/classcv_1_1Tracker.html.

[19] OpenCV, TrackerKCF,https://docs.opencv.org/3.4.1/d2/dff/classcv_1_1TrackerKCF.html.

[20] Wikipedia, Regresión Lineal Simple, https://es.wikipedia.org/wiki/Regresión_lineal.