Detección de bordes

© 2023- F.J. Madrid Cuevas (fjmadrid@uco.es). Universidad de Córdoba. España.

Objetivos

- Obtener el gradiente lumínico en una escena.
- Calcular percentiles de una distribución de valores.
- Detectar bordes.
- Calcular métricas usando la matriz de confusión.

Descripción

En esta práctica vamos a trabajar una tarea de análisis de imágenes conicidad como detección de bordes.

Denominamos borde ("*edge*") en una imagen a la localización de un frontera entre distintas áreas de la misma, por ejemplo en la imagen de la Figura 1, la transición entre el cielo y las montañas o la transición entre la nieve y las raquetas.

El gradiente lumínico suele ser una característica de la imagen que permite detectar estas fronteras, ya que tendrá valores altos de su magnitud en las zonas de transición y valores bajos dentro de las áreas con valores lumínicos más o menos constantes.

Para calcular el gradiente lumínico obtendremos la aproximación de las derivadas parciales Δx y Δy. Estas derivadas parciales las podemos calcular mediante filtrado lineal con el filtro de Sobel.

Hay dos aspectos importantes a tener en cuenta en la detección de bordes: el ruido y la escala de detección.

El ruido afecta mucho a la respuesta del filtro de Sobel por lo que suele ser necesario atenuar dicho ruido en alguna medida.

La escala de detección tiene que ver con la extensión espacial que cubre la transición entre las fronteras. Imagina una cortina con tramado fino. Tendremos bordes a pequeña escala, definidos por dicho tramado, y bordes a mayor escala, definidos por la cortina y la pared sobre la que está montada la cortina.

Para proporcionar mayor robustez al ruido y controlar la escala de detección se puede realizar un filtrado gaussiano previo al cálculo de las derivadas. El valor sigma de la gaussiana controlará la escala de detección. A mayor valor de sigma (o mayor radio) del filtro gaussiano más robustez al ruido y los bordes detectados serán de mayor escala (ver Fig. 1).

Así el proceso para calcular la magnitud del gradiente será:

1. Si el radio ‘r’ del filtro gaussiano es mayor que cero, filtrar la imagen de entrada con un filtro gaussiano de tamaño 2r+1 (cv::GaussianBlur)
2. Calcular las derivadas parciales Δx, Δy usando el filtro de Sobel (cv::Sobel).
3. Calcular la magnitud del gradiente $G = \sqrt{(\Delta^2{}_x + \Delta^2{}_y)}$ (cv::magnitude).

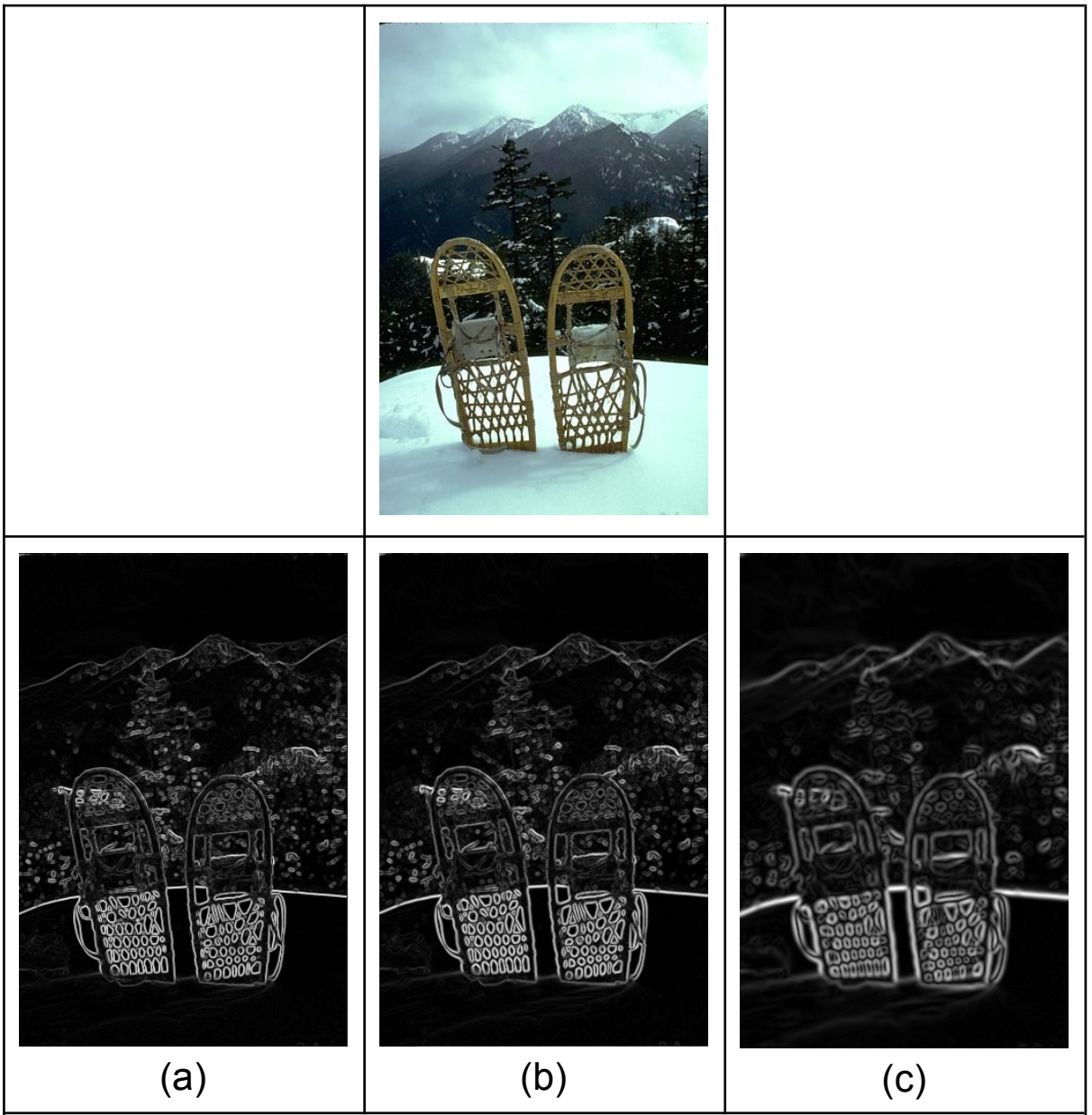


Figura 1. Gradientes obtenidos con radio del filtro gaussiano r=0 (a), r=1 (b) y r=5 (c). Observar cómo a medida que aumenta el radio del filtro gaussiano se pierden detalles finos.

Detectores

Podemos detectar dónde están las fronteras utilizando la magnitud del gradiente, de forma que si un píxel tiene un valor de gradiente suficientemente alto, será clasificado como un píxel frontera o "borde".

El problema ahora es determinar qué valor de magnitud del gradiente es el apropiado para clasificar los píxeles. Además, sería muy difícil encontrar un valor de magnitud que sirva para todas las imágenes, ya que el rango de valores del gradiente no será el mismo en todas ellas.

En esta práctica vamos a utilizar tres alternativas.

Detector usando percentil

Este detector utiliza la hipótesis de que sabemos a priori el porcentaje de píxeles "borde" que pueden haber en la imagen. Por ejemplo, si asumimos que son alrededor de un 20%:

1. Calcularemos el histograma de valores de gradiente (cv::calcHist) en el rango [0, maxG) siendo maxG el máximo valor de magnitud del gradiente (cv::minMaxLoc).
2. Con el histograma, calculamos el valor de magnitud de gradiente `th` que representa el percentil 100%-20% = 80% del mismo.
3. Usamos este valor de magnitud de gradiente `th` para clasificar los píxeles "borde" que serán aquellos con magnitud de gradiente mayor a dicho percentil.

Detector usando Otsu

El problema del método basado en percentil es que se basa en el conocimiento a priori del porcentaje de píxeles "borde" en una imagen, lo cual es mucho suponer.

El método de Otsu permite buscar en un histograma, de forma automática, un valor umbral `th` que divida el histograma en dos clases, que para nosotros esas clases serán "no borde" (<=`th`) y "borde" (> `th`) (cv::threshold).

Detector de Canny

El detector de Canny (cv::Canny) es reconocido como uno de los mejores detectores de bordes. Este detector combina varios procesos como son: la atenuación del ruido y selección de escala de los bordes, la mejora en la localización de los bordes usando un proceso conocido como "supresión de los no-máximos" y una doble umbralización conocida como "histéresis" que utiliza dos umbrales `th1` < `th2`.

El proceso de histéresis divide el histograma de magnitud del gradiente en tres partes "no bode seguro" < `th1`, "borde seguro" > `th2` y "dudosos" que serían los intermedios [`th1`, `th2`]. Los pixeles dudosos, serán clasificados como “borde” si tiene algún ocho-vecino clasificado como “borde seguro”. Este proceso es iterativo hasta que ningún pixel dudoso tenga un ocho-vecino clasificado como “borde”. Nótese que cuando un pixel “dudoso” se clasifica como “seguro”, esto arrastra a otros pixeles “dudosos” vecinos suyos a ser a su vez

clasificados como “seguros”. Cuando el proceso termine, los píxeles “dudosos” que queden serán clasificados como “no borde”.

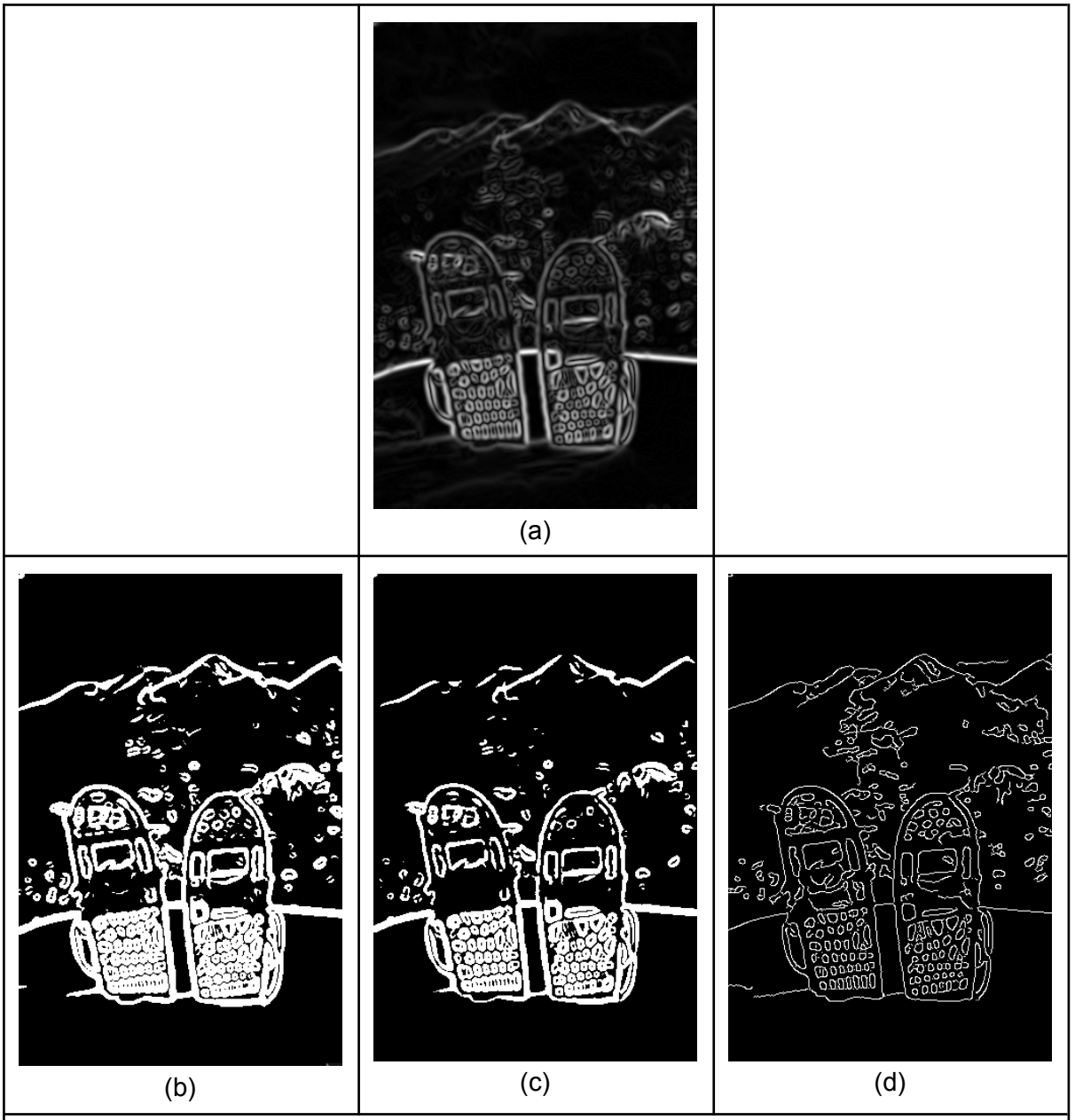


Figura 2. Ejemplos de detecciones. a) Gradiente utilizado, con apertura-sobel = 0 y radio gaussiano r=5. b) Detector percentile con th=0.8, c) Detector Otsu y d) Detector Canny con th1=0.2 y th2=0.8. Nótese que al no contar los métodos “percentile” y “otsu” con la “supresión de no máximos” los bordes son más gruesos que los obtenidos por Canny que sí lo usa.

Métricas

Debemos evaluar el desempeño de nuestro clasificador. Para ello vamos a construir una matriz de confusión y a partir de ella calcularemos varias métricas que nos sirven para evaluar el detector y compararlo con otros detectores.

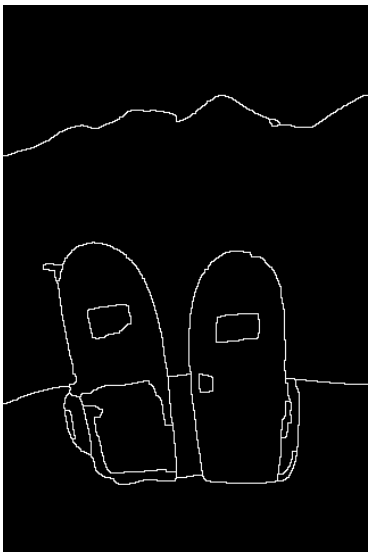
Matriz de confusión

El primer paso para calcular las métricas será construir la matriz de Confusión. Según [Wikipedia]: "*una matriz de confusión es una herramienta que permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Uno de los beneficios de las matrices de confusión es que facilitan ver si el sistema está confundiendo dos clases.*"

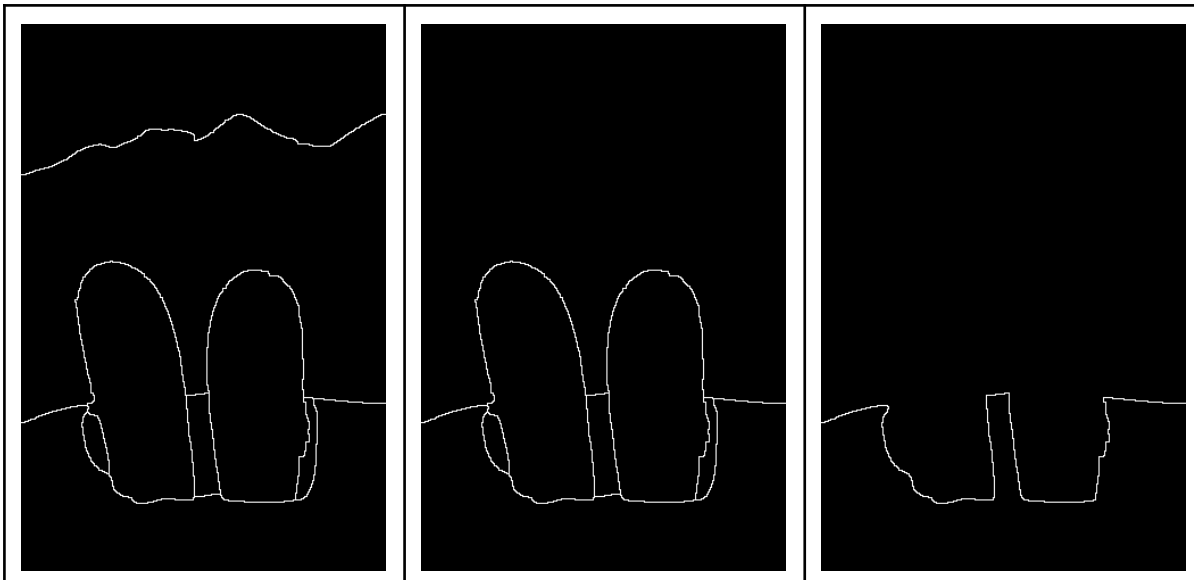
Como el concepto "borde" es subjetivo, si presentamos la misma imagen a un grupo de expertos, en general, cada uno generará una imagen GT distinta. Asumiendo que el valor "0" significa "no borde" y el valor "1" significa "borde" si sumamos las distintas imágenes GT generadas por los expertos obtendremos una imagen de consenso "C" donde los píxeles con mayor valor ("votos") indicarán píxeles donde el consenso de todos los expertos a que es ese píxel es "borde" es mayor. Así a partir de una imagen de consenso "C" normalizada MIN-MAX al rango (0%, 100%) podemos generar una imagen de GT para un valor mínimo de consenso "c" umbralizándola, es decir $GT = C >= c$.

(a)

(b)



(c)



(d) (e) (f)

Figura 3: Generación de la imagen “Ground-Truth” (GT) a partir de la imagen de consenso. (a) Imagen original, (b) imagen de consenso generada como la suma de las imágenes GT dadas por los expertos. Observar cómo los valores más claros (mayor número de votos) marcan píxeles con mayor confianza de que sean “bordes”. (c), (d) (e) y (f) distintas imágenes GT generadas umbralizando la imagen de consenso con un valor mínimo de consenso del 25%, 50%, 75% y 100% de los votos respectivamente.

Métricas Sensibilidad, Precisión y F1

A partir de la matriz de confusión vamos a calcular (ver [aquí] cómo) tres métricas:

Sensibilidad: Mide qué porcentaje de píxeles que realmente son borde han sido detectados por el nuestro detector. Cuanto más alto mejor.

Precisión: Mide qué porcentaje de píxeles que nuestro detector dice que son borde, realmente lo son. Cuanto más alto mejor.

F1: Es una métrica que mide el balance entre *Sensibilidad* y *Precisión* usando la media armónica entre ambas. Cuanto más alto mejor.

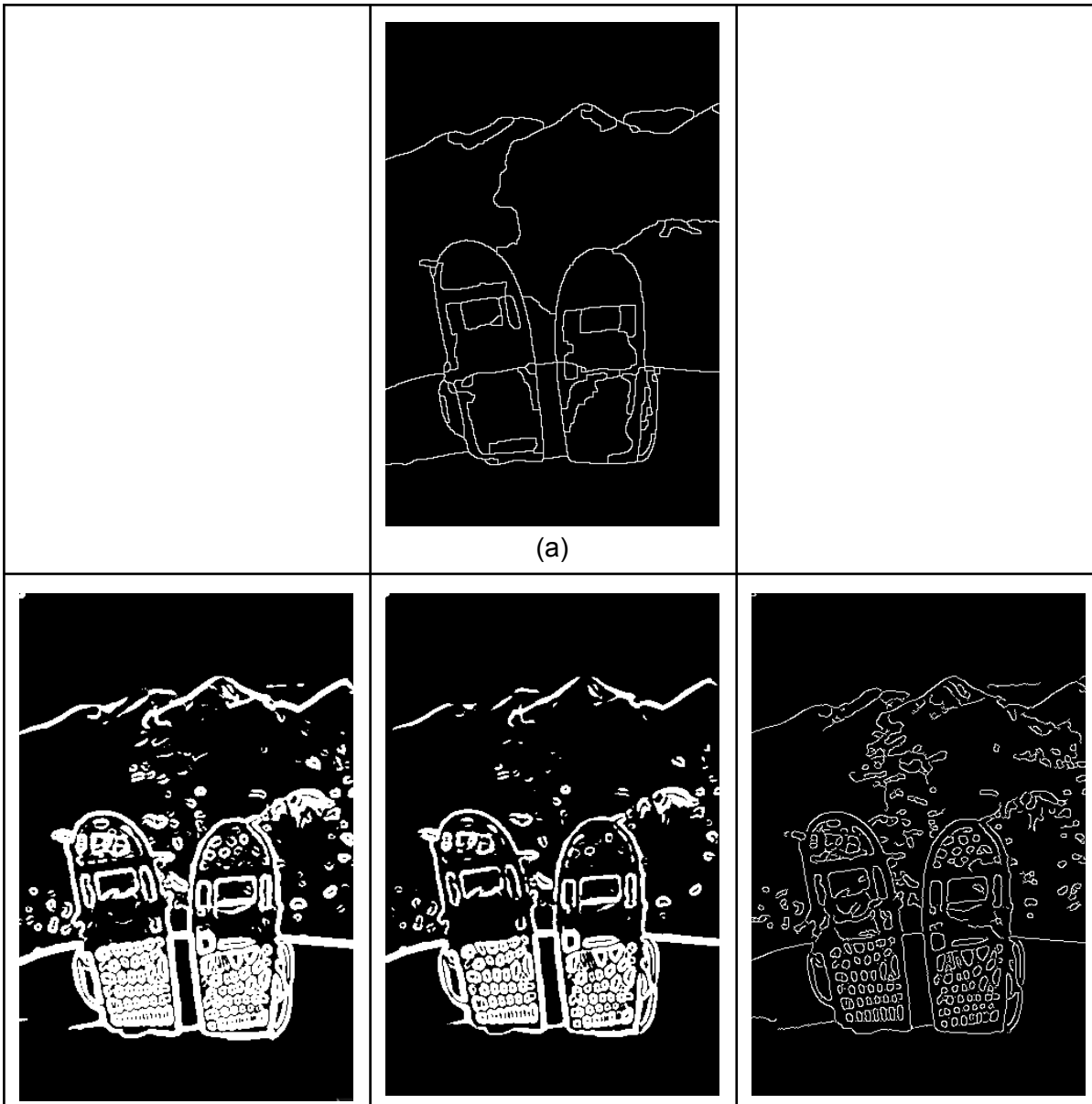


Figura 4. Evaluación de las detecciones. a) Ground-truth obtenido por consenso de al menos el 10% b) Detector percentile c) Detector Otsu y d) Detector Canny. Los parámetros utilizados son los mismos de la Figura 2.

Evaluación

Hay que desarrollar un programa que implemente los tres métodos descritos para detectar bordes usando la magnitud del gradiente como característica. El programa debe mostrar las métricas Sensibilidad, Precisión y coeficiente F1 obtenidas por el método utilizado.

Funcionalidad	**Puntos**
test_common_code	hasta 5
Descripción del código realizado en el vídeo.	hasta 2.5
Prueba del programa edge_detector con las imágenes 2018.jpg 2018_gt.png explicando cómo afecta los distintos parámetros	hasta 2.5