

Trabajo Practico 2

Luis Fernando Recalde

Instrucciones: Utilizar la imágenes utilizadas en el practico anterior.

1. Resolver

- (a) Con los parámetros obtenidos de la calibración de la cámara usar función OpenCV `cv2.undistort()`, los resultados se muestran en Fig. 1

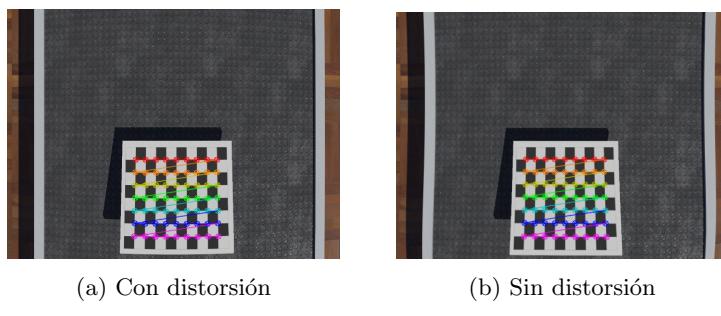


Figure 1: Corrección de imágenes usando la función `cv2.undistort()`

- (b) Usando las ecuaciones, implementar máscaras de convolución de 3×3 , 5×5 , y 7×7 para realizar filtrado pasa-bajos (media y Gaussiano). Implementar un filtro no lineal de mediana de 3×3 y 5×5 . Comparar con las funciones de OpenCV.

- (1) Filtro paso bajos media usando convoluciones 3×3 , 5×5 , y 7×7 .

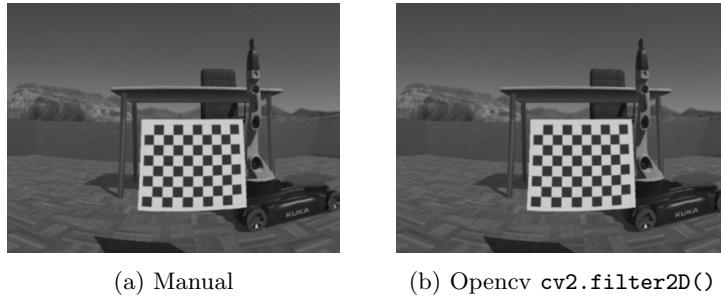
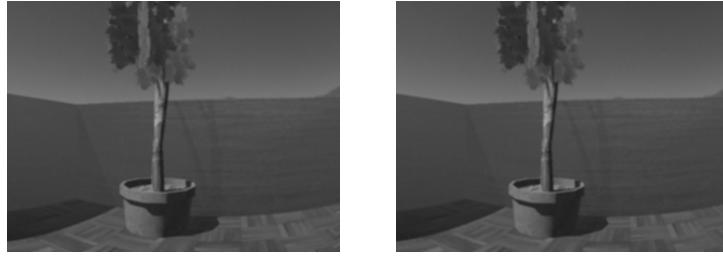


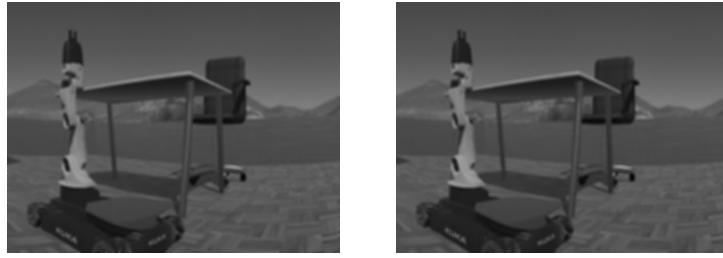
Figure 2: Filtro media con convolución 3×3

Los resultados de este filtro media usando una convolución 3×3 se muestran en Fig. 2, el resultado de aplicar una mascara de 5×5 se muestra en Fig. 3. Finalmente el resultados aplicando una mascara de 7×7 se muestra en Fig. 4, el tiempo de computo es un factor a considerar mientras mayor la mascara de convolución el tiempo de computo aumenta.

- (2) Filtro paso bajos Gaussiano usando convoluciones 3×3 , 5×5 , y 7×7 , el valor considerado es de $\sigma = 1$.



(a) Manual

(b) Opencv `cv2.filter2D()`Figure 3: Filtro media con convolución 5×5 

(a) Manual

(b) Opencv `cv2.filter2D()`Figure 4: Filtro media con convolución 7×7

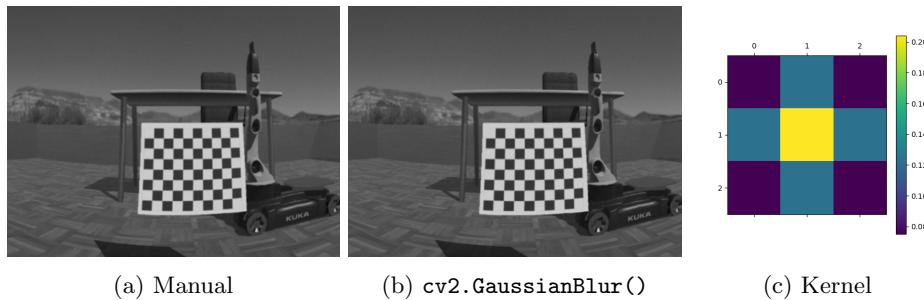
Los resultados de este filtro Gaussiano usando una convolución 3×3 se presentan en Fig. 5, el resultado de aplicar una mascara de 5×5 se muestra en Fig. 6.

Finalmente el resultados aplicando una mascara de 7×7 se muestra en Fig. 7.

Además se presenta la matriz que se aplico a cada kernel donde se aprecia el comportamiento de una función gaussiana bidimensional.

- (3) Filtro paso bajos mediana usando convoluciones 3×3 , 5×5 , y 7×7 .

A continuación se aplica un filtro no lineal, los resultados aplicando una mascara de 3×3 se muestran en Fig. 8.



(a) Manual

(b) `cv2.GaussianBlur()`

(c) Kernel

Figure 5: Filtro Gaussiano con convolución 3×3

Los resultados aplicando una mascara de convolución de 5×5 se presenta en la Fig. 9. Finalmente la imagen obtenida aplicando un filtro mediana no lineal con un kernel 7×7

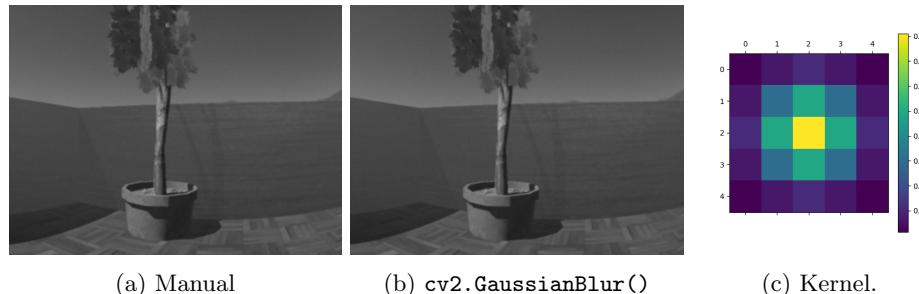


Figure 6: Filtro Gaussiano con convolución 5×5

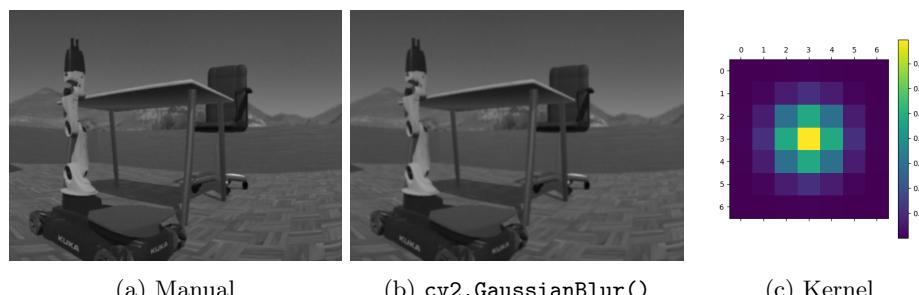


Figure 7: Filtro Gaussiano con convolución $5x5$

se presenta Fig. 10, un aspecto importante de este filtro es su comportamiento no lineal, lo cual llega a ser útil al momento de reducir ruido en las imágenes.

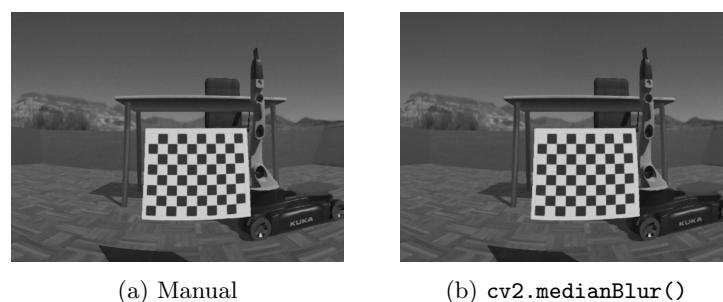
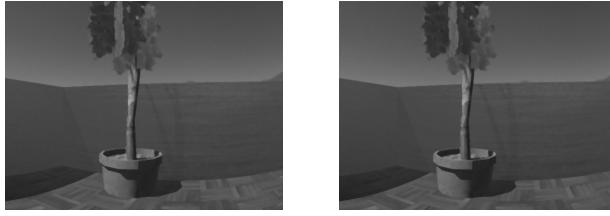
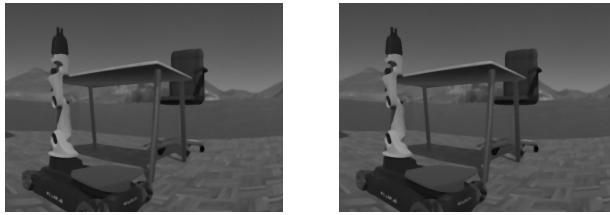
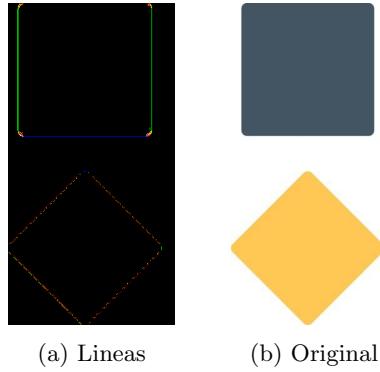


Figure 8: Filtro mediana con convolución 3×3

(a) Manual (b) `cv2.medianBlur()`Figure 9: Filtro mediana con convolución 5×5 (a) Manual (b) `cv2.medianBlur()`Figure 10: Filtro mediana con convolución 7×7

- (c) Usar máscaras de convolución para realizar la detección de líneas horizontal, vertical, -45° y 45° . La imagen de salida tiene que tener color azul para línea horizontal, verde para vertical,



(a) Lines (b) Original

Figure 11: Filtro mediana con convolución 7×7

rojo para -45° y amarillo para 45° . Los resultados de esta operación se muestran en Fig. 11, se aprecia como se logra identificar cada linea, y asignar los colores respectivos, esta operación se puede hacer mas robusta utilizando directamente el ángulo del gradiente y así seleccionar las lineas que estén dentro de cierto rango.

- (d) Realizar el cálculo del gradiente de las imágenes en X (G_x) e Y (G_y). Mostrar las imágenes G_x , G_y y $G_x + G_y$. Encontrar el módulo y el ángulo del gradiente en alguna zona que considere de interés de la imagen y mostrar estos valores.

Los resultados de los gradientes en los ejes x y y se muestran en Fig. 12b y Fig. 12a respectivamente, además la suma de estos valores se encuentra en Fig. 12c.

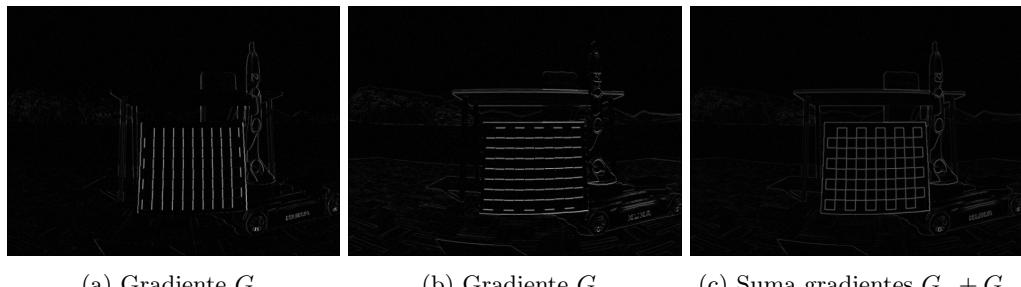


Figure 12: Calculo del gradiente.

Finalmente los resultados del ángulo en la región de interés se muestra en Fig. 13.

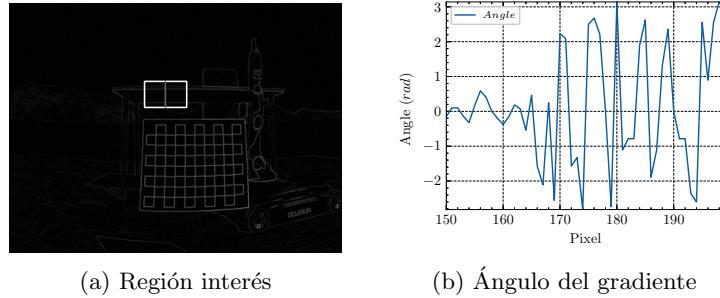


Figure 13: Calculo del ángulo en una región de interés

- (e) Usando las ecuaciones, implementar máscaras de convolución para realizar el filtrado pasa-altos (Robert, Prewitt, Sobel y Frei-Chen). Comparar con las funciones de OpenCV

 - (1) Filtro pasa-altos Robert, los resultados de este proceso se encuentran en Fig. 14, se considera ala dimensión de la mascara de 3.

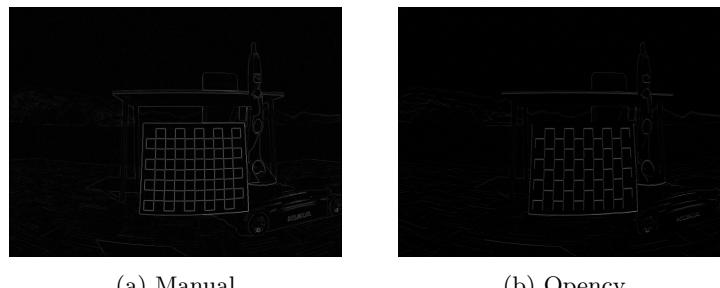


Figure 14: Pasa-altos Robert

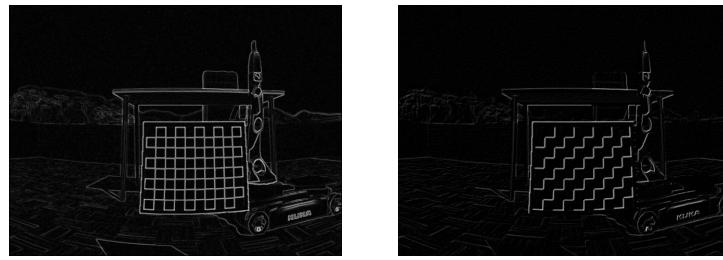


Figure 15: Pasa-altos Prewitt.

- (2) Filtro pasa-altos Prewitt, el resultado de este método se encuentra en Fig. 15. En los resultados se puede apreciar que no se obtuvieron las mismas imágenes, lo que puede ser efecto de que no se conoce que matrices usa Opencv en este procedimiento.
 - (3) Filtro pasa- altos usando el Sobel, para estos resultados se aplico respecto a los ejes x e y , los resultados con respecto al eje y se encuentran en Fig. 16, de igual forma se utilizaron las ecuaciones y la función propuesta por Opencv.

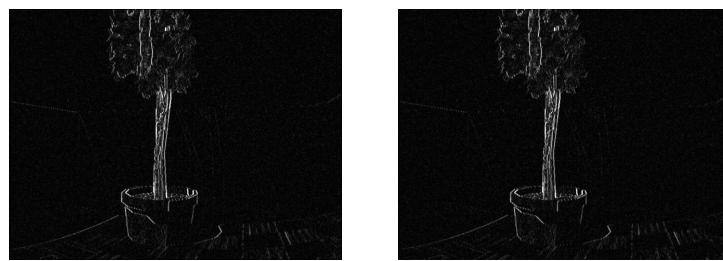


Figure 16: Pasa-altos Sobel y

Los resultados en el eje x , estos se presentan en Fig. 17, usando las formulas o la función se llego al mismo resultado.

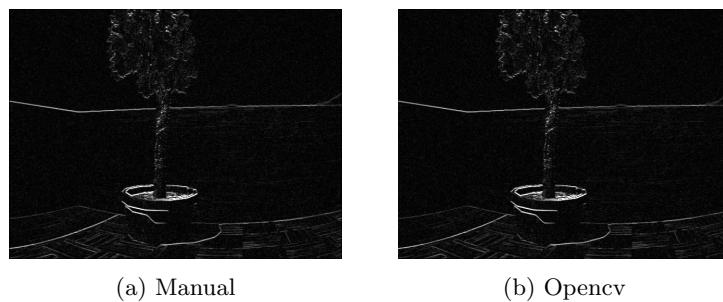


Figure 17: Pasa-altos Sobel x .

- (4) Filtro pasa-altos usando el Frei-Chen, para este algoritmo solo se dispone de las formulas

ya que Opencv no dispone de una función para esta operación, el resultado de esta operación respecto a x e y se encuentra en Fig. 18

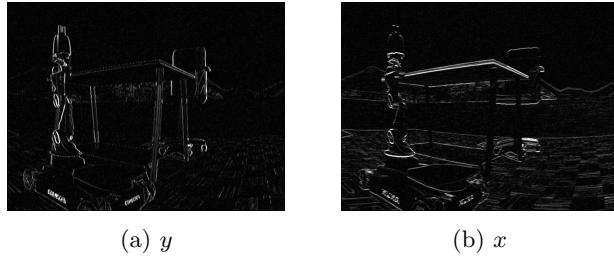


Figure 18: Pasa-altos Frei-Chen.

- (f) Calcular el Laplaciano de las imágenes y usarlo para realzar las imágenes.

El resultado de este algoritmo se encuentra en Fig. 19, además se logra apreciar como se logra resaltar los detalles de las imágenes.

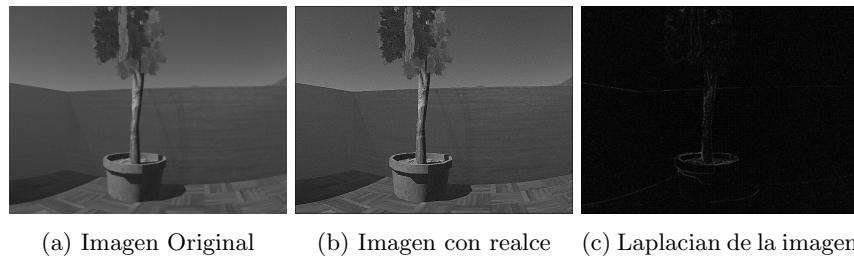


Figure 19: Calculo Laplaciano.

- (g) Implementar un filtrado High Boost. Una imagen filtrada de paso alto puede ser calculada como la diferencia entre la original y una versión de esta imagen que ha pasado por un filtro de paso bajo:

- (1) Para este algoritmo se utiliza $H_f = \text{Original} - L_f$, el resultado se encuentra en Fig. 20.

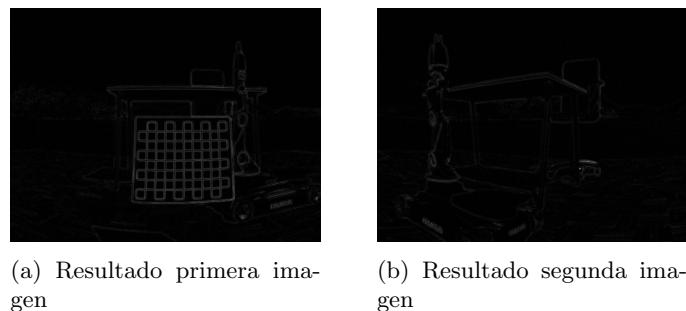
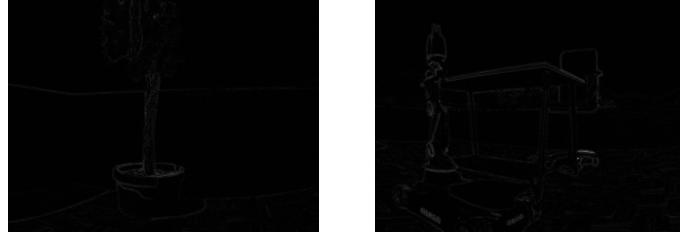


Figure 20: Filtro pasa-alto usando paso-bajo

- (2) Multiplicando la imagen original por un factor de amplificación, a , se obtiene la definición de un filtro high-boost o de énfasis de las altas frecuencias, la ecuación a aplicar es la siguiente $H_f = a \text{ Original} - L_f$, con los resultados en Fig. 21.

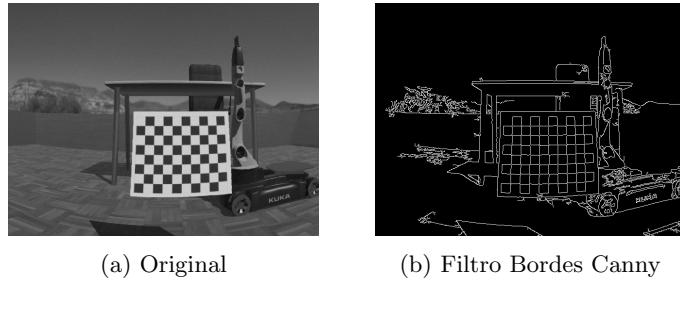


(a) Segunda imagen

(b) Tercera imagen

Figure 21: Filtro pasa-alto usando paso-bajo

- (h) Usar la función de OpenCV para realizar el filtrado de Canny, los resultados de este filtro se presentan en Fig. 22.



(a) Original

(b) Filtro Bordes Canny

Figure 22: Filtro Canny

- (i) Implementar la Diferencia de Gaussianas (DoG) para diferentes valores de σ . Para este filtro se utilizó los valores de $\sigma = 1.5$ y $\sigma = 0.9$, estos valores se pueden ver en Fig. 23. Finalmente

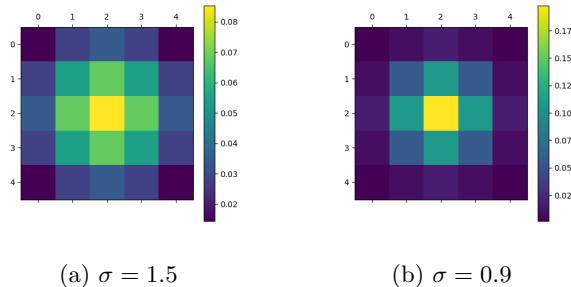
(a) $\sigma = 1.5$ (b) $\sigma = 0.9$

Figure 23: Mascaras convolución utilizadas

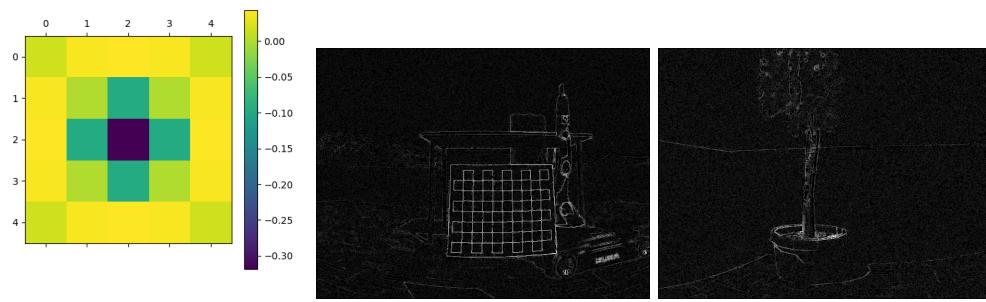
el resultado de este filtro se observa en Fig. 24.



(a) Diferencia

Figure 24: Diferencia Gaussiana

- (j) Implementar la ecuación del Laplaciano del Gaussiano (LoG). Los resultados de este filtro se encuentran en Fig. 25b y Fig. 25c, además se muestra el kernel aplicado a este algoritmo Fig. 25a.



(a) Kernel Laplaciano Gauss

(b) Primera Imagen

(c) Segunda Imagen

Figure 25: Laplaciano Gaussiano

Este trabajo puede ser descargado en(Github).