รูปภาพ Smoothing

เป้าหมาย

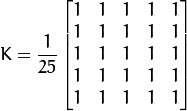
เรียนรู้เพื่อ:

* เบลอภาพด้วยตัวกรองความถี่ต่ำ ๆ
* ใช้ตัวกรองที่กำหนดเองกับภาพ (2D convolution)

2D Convolution (การกรองภาพ)

เช่นเดียวกับสัญญาณภาพแบบ 1 มิติภาพสามารถกรองด้วยตัวกรอง Low Pass (LPF), ตัวกรองผ่านสูง (HPF) ฯลฯ LPF ช่วยในการลบเสียงรบกวนภาพ ฯลฯ ตัวกรอง HPF ช่วยในการค้นหาขอบใน ภาพ

OpenCV มีฟังก์ชัน**cv2.filter2D ()** เพื่อรวบรวมเคอร์เนลด้วยรูปภาพ ตัวอย่างเช่นเราจะลองใช้ตัวกรองเฉลี่ยสำหรับรูปภาพ เคอร์เนลตัวกรองเฉลี่ย 5x5 จะมีลักษณะดังนี้:



การดำเนินการเป็นเช่นนี้: เก็บเคอร์เนลนี้ไว้เหนือพิกเซลเพิ่ม 25 พิกเซลทั้งหมดที่อยู่ใต้เคอร์เนลนี้ใช้ค่าเฉลี่ยและแทนที่ pixel กลางด้วยค่าเฉลี่ยใหม่ มันยังคงดำเนินการนี้สำหรับพิกเซลทั้งหมดในภาพ ลองใช้รหัสนี้และตรวจสอบผล:

**import** **cv2**

**import** **numpy** **as** **np**

**from** **matplotlib** **import** pyplot **as** plt

img = cv2.imread('opencv\_logo.png')

kernel = np.ones((5,5),np.float32)/25

dst = cv2.filter2D(img,-1,kernel)

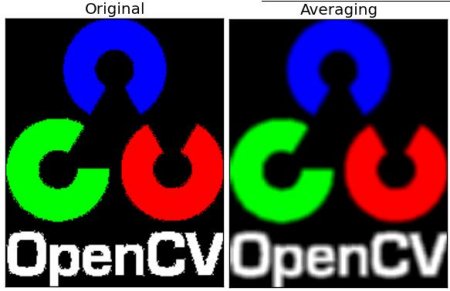
plt.subplot(121),plt.imshow(img),plt.title('Original')

plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(dst),plt.title('Averaging')

plt.xticks([]), plt.yticks([])

plt.show()

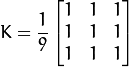


การทำให้ภาพพร่ามัว (Image Smoothing)

การทำให้ภาพพร่ามัวทำได้โดยการหมุนภาพด้วยเคอร์เนลกรองต่ำ จะเป็นประโยชน์ในการถอดเสียงรบกวน จะลบเนื้อหาความถี่สูง (เช่นเสียงรบกวน, ขอบ) ออกจากภาพ ดังนั้นขอบจะเบลอเล็กน้อยในการดำเนินการนี้ (มีเทคนิคการทำให้เปรอะเปื้อนที่ไม่เบลอขอบเกินไป) OpenCV มีเทคนิคการทำให้เบลอเป็นหลักสี่ประเภท

1. เฉลี่ย

นี้จะกระทำโดย convolving ภาพที่มีตัวกรองกล่อง normalized มันใช้เวลาเฉลี่ยของพิกเซลทั้งหมดภายใต้พื้นที่เคอร์เนลและแทนที่องค์ประกอบกลาง นี้จะกระทำโดยฟังก์ชั่น **cv2.blur ()** หรือ **cv2.boxFilter**() ตรวจสอบเอกสารสำหรับรายละเอียดเพิ่มเติมเกี่ยวกับเคอร์เนล เราควรระบุความกว้างและความสูงของเคอร์เนล ตัวกรองกล่องจดหมายปกติ 3x3 จะมีลักษณะดังนี้:



**บันทึก :** หากคุณไม่ต้องการที่จะใช้ตัวกรองกล่องปกติใช้**cv2.boxFilter**() ส่งอาร์กิวเมนต์normalize = เท็จไปยังฟังก์ชัน

ตรวจสอบตัวอย่างด้านล่างที่มีเคอร์เนลขนาด 5x5:

**import** **cv2**

**import** **numpy** **as** **np**

**from** **matplotlib** **import** pyplot **as** plt

img = cv2.imread('opencv\_logo.png')

blur = cv2.blur(img,(5,5))

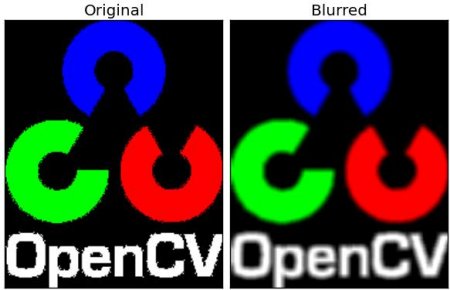
plt.subplot(121),plt.imshow(img),plt.title('Original')

plt.xticks([]), plt.yticks([])

plt.subplot(122),plt.imshow(blur),plt.title('Blurred')

plt.xticks([]), plt.yticks([])

plt.show()



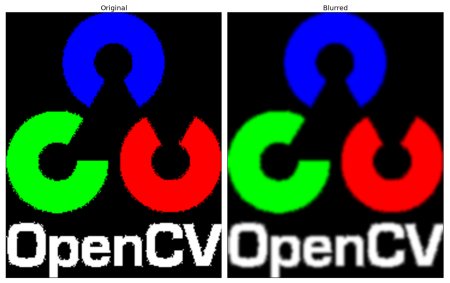
2. Gaussian Blurring

ในกรณีนี้ใช้ตัวกรองแบบเกาส์แทนที่จะใช้ตัวกรองช่อง มันจะทำกับฟังก์ชั่น **cv2.GaussianBlur**() เราควรระบุความกว้างและความสูงของเคอร์เนลซึ่งควรเป็นบวกและคี่ นอกจากนี้เราควรระบุส่วนเบี่ยงเบนมาตรฐานในทิศทาง X และ Y sigmaX และ sigmaY ตามลำดับ ถ้ามีการระบุ sigmaX เท่านั้น sigmaY จะถูกใช้เหมือนกับ sigmaX ถ้าทั้งสองได้รับเป็นศูนย์จะคำนวณจากขนาดเคอร์เนล การทำให้เปรอะเปื้อนแบบ Gaussian มีประสิทธิภาพในการกำจัดสัญญาณรบกวนแบบเกาส์จากภาพ

ถ้าคุณต้องการคุณสามารถสร้างเคอร์เนล Gaussian กับฟังก์ชั่น **cv2.getGaussianKernel**()

โค้ดข้างต้นสามารถแก้ไขได้สำหรับการทำให้ Gaussian blurring:

blur = cv2.GaussianBlur(img,(5,5),0)

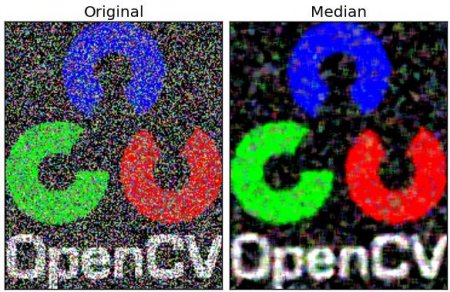


3. การเบลอกลางสี่

ที่นี่ฟังก์ชัน**cv2.medianBlur ()**ใช้ค่ามัธยฐานของพิกเซลทั้งหมดภายใต้พื้นที่เคอร์เนลและองค์ประกอบส่วนกลางจะถูกแทนที่ด้วยค่ามัธยฐานนี้ นี้มีประสิทธิภาพสูงต่อเสียงจากเกลือและพริกไทยในภาพ สิ่งที่น่าสนใจคือในตัวกรองด้านบนองค์ประกอบส่วนกลางเป็นค่าที่คำนวณใหม่ซึ่งอาจเป็นค่าพิกเซลในภาพหรือเป็นค่าใหม่ แต่ในการทำให้เบลอกลางมัธยฐานองค์ประกอบส่วนกลางจะถูกแทนที่ด้วยค่าพิกเซลบางส่วนในรูปภาพ จะช่วยลดเสียงรบกวนได้อย่างมีประสิทธิภาพ ขนาดของเคอร์เนลควรเป็นจำนวนเต็มคี่ที่เป็นบวก

ในการสาธิตนี้ผมได้เพิ่มเสียงรบกวน 50% ในภาพต้นฉบับของเราและใช้การเบลอแบบมัธยฐาน ตรวจสอบผลลัพธ์:

median = cv2.medianBlur(img,5)



4. การกรองแบบทวิภาคี

**cv2.bilateralFilter ()** มีประสิทธิภาพในการกำจัดเสียงรบกวนในขณะที่รักษาขอบคม แต่การดำเนินการจะช้ากว่าตัวกรองอื่น ๆ เราได้เห็นว่าตัวกรอง gaussian ใช้เวลาละแวกใกล้เคียงพิกเซลและหาค่าเฉลี่ยถ่วงน้ำหนักของ gaussian ตัวกรองแบบเกาส์นี้เป็นฟังก์ชันหนึ่งของพื้นที่เพียงอย่างเดียวนั่นคือพิกเซลในบริเวณใกล้เคียงถือว่าอยู่ในระหว่างการกรอง ไม่ได้พิจารณาว่าพิกเซลมีความเข้มเท่ากันหรือไม่ ไม่พิจารณาว่าพิกเซลเป็นพิกเซลขอบหรือไม่ ดังนั้นมันจึง blurs ขอบที่เราไม่ต้องการที่จะทำ

ไส้กรองทวิภาคียังใช้ตัวกรอง gaussian ในอวกาศ แต่มีอีกหนึ่งตัวกรอง gaussian ซึ่งเป็นฟังก์ชันของความแตกต่างของพิกเซล ฟังก์ชัน Gaussian ของพื้นที่ตรวจสอบให้แน่ใจว่าเฉพาะพิกเซลใกล้เคียงเท่านั้นที่จะถือว่าเป็นภาพเบลอขณะที่ฟังก์ชัน gaussian ของความแตกต่างของความเข้มให้แน่ใจว่าเฉพาะพิกเซลเหล่านี้ที่มีความเข้มใกล้เคียงกับพิกเซลกลางเท่านั้นจึงจะถือว่าเป็นภาพเบลอ ดังนั้นจึงรักษาขอบเนื่องจากพิกเซลที่ขอบจะมีรูปแบบความเข้มขนาดใหญ่

ตัวอย่างด้านล่างแสดงการใช้ตัวกรองแบบทวิภาคี (สำหรับรายละเอียดเกี่ยวกับอาร์กิวเมนต์ให้ไปที่เอกสาร)

blur = cv2.bilateralFilter(img,9,75,75)

