Handbag Logo Segmentation | Final Project

Introduction

ปัญหาการค้นหา Logo ของ Brand จากภาพกระเป๋า Brand-named ที่มีตำแหน่ง สี และองค์ประกอบต่างๆของภาพแตกต่างกันทั้งหมด 6 ภาพ 6 Brand โดยใช้ Digital Image Processing Technique ในการหาตำแหน่งของ Logo และเปลี่ยนบริเวณที่ไม่ใช้ Logo เป็นสีพื้น ในที่นี้ได้เลือกเปลี่ยนเป็นสีขาว

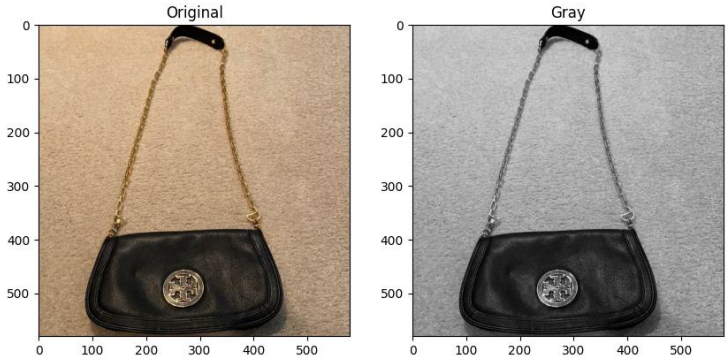
Method

Library ที่เลือกใช้งานคือ OpenCV โดยมีการใช้ Built-in Method ภายใน Library ในการ Process รูปภาพ
วิธีที่ได้เลือกใช้ในการทำ Image Processing ในครั้งนี้ประกอบไปด้วย

1. **Color Space Convert**

   เนื่องจาก Dataset ภาพที่มีนั้นเป็นภาพแบบสีแต่เมื่อพิจารณาปัจจัยต่าง ๆ ในการทำ Logo Segmentation และได้มีการลองทำ Segmentation แยกทั้ง 3 Channel พบว่าไม่มีความจำเป็นที่จะต้องใช้ภาพสีในการทำ จึงได้ทำการแปลงจาก Color Image เป็น Grayscale Image ทั้งหมด

   ตัวอย่างผลลัพธ์



   จากตัวอย่างผลลัพธ์ที่ได้จากใช้งาน Method cvtColor โดยมีการส่งค่า Parameter รูปภาพเดิมและ Function การเปลี่ยนปริภูมิสีเป็น cv.COLOR_BGR2GRAY[1]

ตัวอย่าง Code

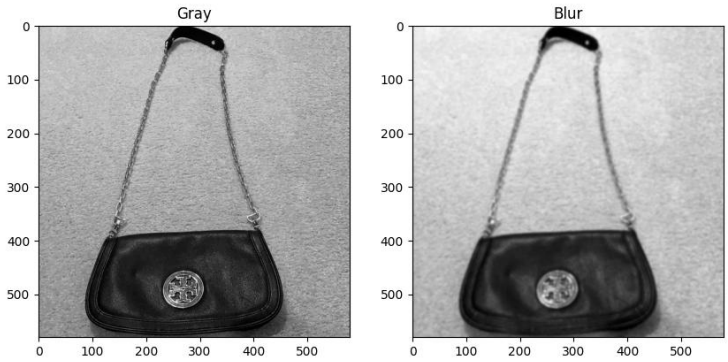
```
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
```

[1] เนื่องจากในการอ่านค่าสีของภาพของ Library OpenCV นั้นจะอ่านค่าเป็น BGR Color space จึงใช้ COLOR_BGR2GRAY

2. Gaussian Blur

การใช้ Gaussian Blur เป็นส่วนหนึ่งในการทำ Process ภาพครั้งนี้มีสาเหตุมาจากว่าภาพบางภาพนั้นมี Noise ทำให้เมื่อนำ Image Thresholding แล้วพบว่า Noise นั้นได้ส่งผลกระทบต่อผลลัพธ์ที่ได้ เพื่อเป็นการลบ Noise รวมถึงลด Detail ที่ไม่จำเป็นออกไปจากภาพจึงมีการใช้งาน Gaussian Blur

ตัวอย่างผลลัพธ์



จากตัวอย่างผลลัพธ์ที่ได้จากการใช้งาน Method GaussianBlur โดยมีการส่งค่า Parameter โดยภาพจะเป็นภาพ Grayscale จากข้อที่ 1 และมีการกำหนดค่าขนาดของ Kernel ในการทำ Gaussian Blur เป็น 9 x 9 สามารถสังเกตุได้ว่ารายละเอียดที่เราไม่ได้ต้องการบริเวณพรมและโซ่ของสายคล้องนั้นได้ลดลง และถ้าหากพิจารณาแล้วคิดว่ายังมีรายละเอียดเยอะอยู่ก็ยังสามารถปรับขนาดของ Kernel ให้ใหญ่กว่านี้ได้อีก

โดยในการทำ Logo Segmentation ครั้งนี้มีการใช้ Kernel ขนาดต่างกันขึ้นอยู่กับรูปภาพโดยมีการใช้ขนาด

- **9 x 9** สำหรับภาพ handbag1, handbag2, handbag 4 และ handbag 8
- **15 x 15** สำหรับภาพ handbag3 และ handbag 7

ตัวอย่าง Code

```
blur = cv.GaussianBlur(gray, (9, 9), 0)
```

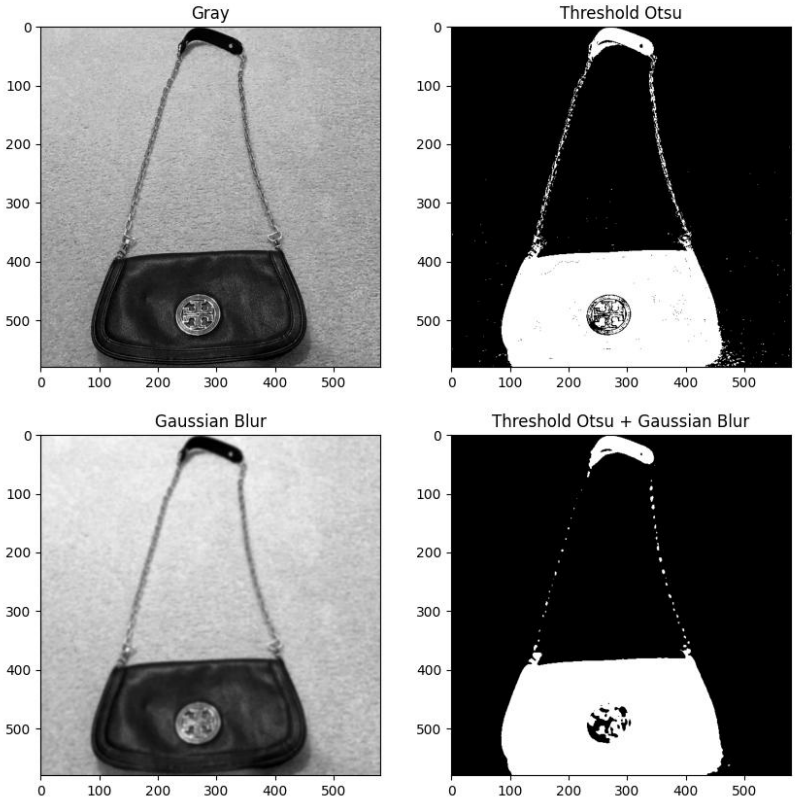
3. Image Thresholding

ใช้ Technique นี้ในการทำ Image Segmentation คร่าว ๆ เพื่อแยกกระเป๋าออกจากพื้นหลัง เพื่อให้สามารถโฟกัสไปยังการค้นหา Logo ของ Brand ที่อยู่บนกระเป๋าได้ดียังขึ้น ในหัวข้อย่อยทั้ง 2 ข้อนี้จะอธิบายว่าทำไมถึงเลือกใช้การทำ Thresholding ทั้ง 2 แบบนี้

3.1. Otsu's Thresholding

Otsu's Thresholding เป็นการทำ Threshold ที่นิยมทำกันเป็นอย่างมากเพื่อแยก Foreground และ Background ออกจากกันวิธีนี้ได้ผลเป็นอย่างมากในการ Scope การหา Logo ลงเหลือเพียงบริเวณบนกระเป๋า

ตัวอย่างผลลัพธ์



ตัวอย่างนี้แสดงให้เห็นถึงความแตกต่างระหว่างการทำ Gaussian Blur ก่อนทำ Thresholding กับไม่ทำ Gaussian Blur ก่อนทำ Thresholding เนื่องจากพื้นหลังของรูปที่นำมาทำ Thresholding นั้นเป็นพรมทำให้เกิด Noise สีขาวบริเวณขอบทางขวาล่างของกระเป๋า และมี Noise สีดำภายในกระเป๋าที่เกิดจากการสะท้อนของหนังกระเป๋าที่ไม่สม่ำเสมอกัน Detail ในส่วนนี้เยอะจนเกินไปทำให้การใช้ Otsu's Thresholding ไม่ดีเท่าที่ควร การทำ Gaussian Blur ก่อนจะช่วยลดรายละเอียดที่ไม่จำเป็นออกไปจะสังเกตได้ว่า Otsu's Thresholding ด้านล่างขวานั้นให้ผลลัพธ์ที่ดีกว่า

ตัวอย่าง Code

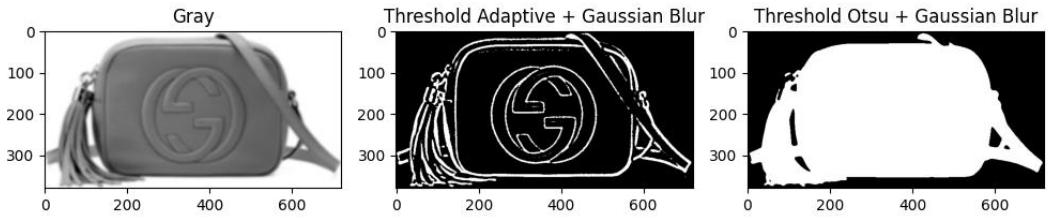
```
ret, thr = cv.threshold(blur, 0, 255, cv.THRESH_BINARY_INV + cv.THRESH_OTSU)
```

ใช้ `threshold` และส่ง Parameter `cv.THRESH_BINARY_INV + cv.THRESH_OTSU` เพื่อระบุว่าจะใช้งาน Otsu's method และผลลัพธ์ที่ได้จะเป็น Binary Inverse Value (Foreground สีขาว และ Background สีดำ)

3.2. Adaptive Gaussian Thresholding

การทำ Adaptive Thresholding เป็นหนึ่งในวิธีที่ใช้ในการแยก Foreground และ Background เช่นเดียวกับ Otsu แต่มีการเลือกใช้ Adaptive Gaussian Thresholding เนื่องจากภาพกระเป๋าบางภาพนั้น Logo มีสีคล้ายคลึงและกลมกลืนไปกับตัวกระเป๋าทำให้ไม่สามารถแยก Logo ออกจากกระเป๋าได้

ตัวอย่างผลลัพธ์



ตัวอย่างแสดงให้เห็นว่าทำไมบางภาพถึงเลือกใช้ Adaptive Gaussian Thresholding แทนการใช้ Otsu's Thresholding สังเกตได้ว่าหากใช้ Thresholding แบบ Otsu รายละเอียดของ Logo นั้นหายไปจึงมีการเลือกใช้ Adaptive Gaussian Thresholding ในบางรูปภาพ

ตัวอย่าง Code

```
thr = cv.adaptiveThreshold(gray, 255, cv.ADAPTIVE_THRESH_GAUSSIAN_C,
cv.THRESH_BINARY_INV, 11, 2)
```

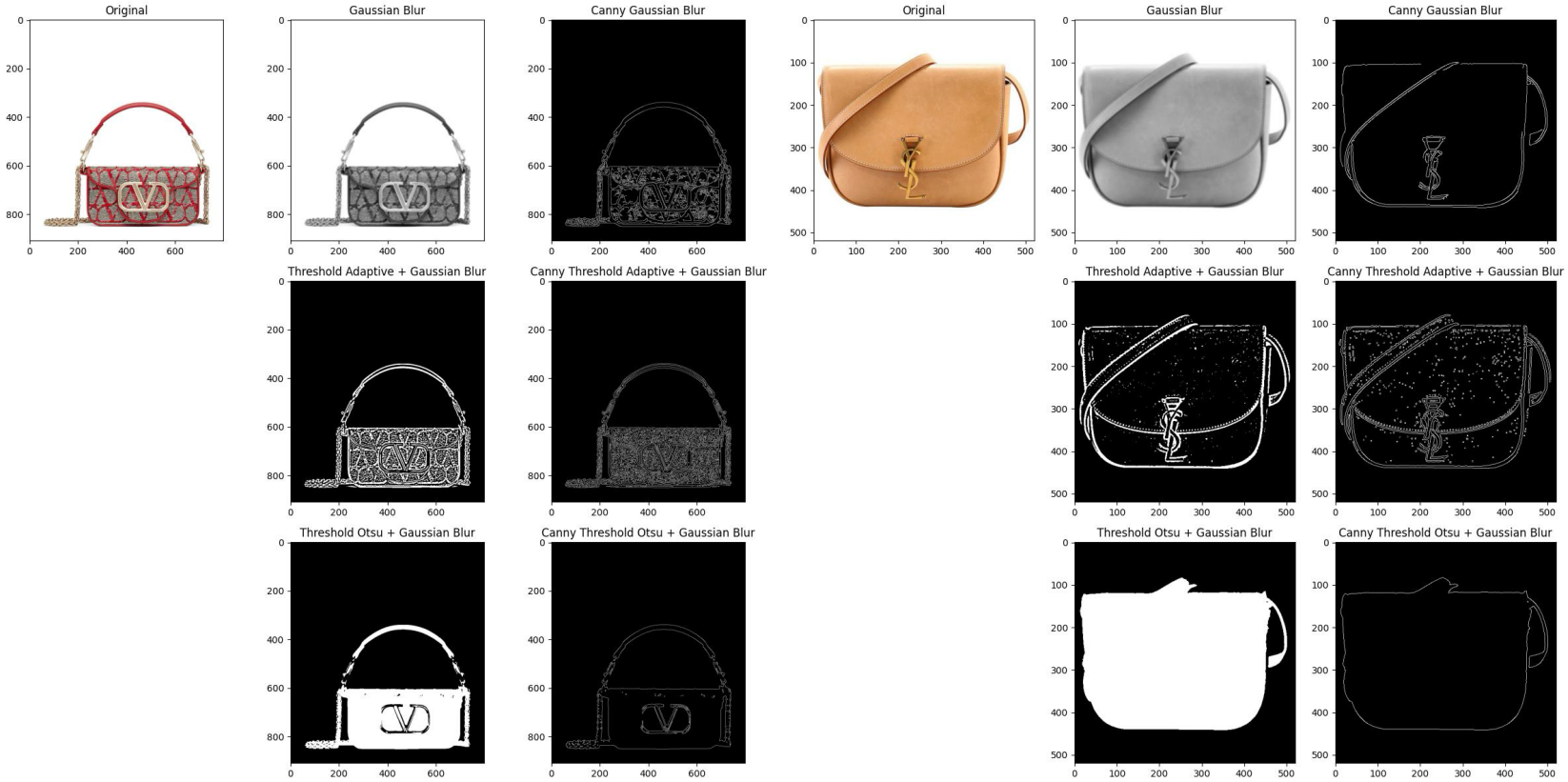
ใช้ `adaptiveThreshold` และส่ง Parameter `cv.ADAPTIVE_THRESH_GAUSSIAN_C, cv.THRESH_BINARY_INV` เพื่อระบุว่าจะใช้งาน Gaussian Thresholding และผลลัพธ์ที่ได้จะเป็น Binary Inverse Value (Foreground สีขาว และ Background สีดำ)

ต่อมาคงมีคำถามว่าทำไมถึงทำ Thresholding ก่อนทำ Canny Edge Detection ซึ่งจะกล่าวถึงในหัวข้อถัดไป ดังนั้นเพื่อให้เค้าโครงของ Logo ของภาพกระเป๋ายังคงอยู่จึงเลือกใช้ Thresholding ที่เหมาะสมกับภาพนั้น ๆ

4. Canny Edge Detector

Canny Edge Detector เป็นหนึ่งใน Algorithm ยอดนิยมสำหรับการทำ Edge Detection และผลลัพธ์จากการใช้ Canny ที่เราอยากได้นั่นคือ Edge ทั้งภายนอก (ขอบของกระเป๋า) และ Edge บริเวณของ Logo บนกระเป๋า

ตัวอย่างผลลัพธ์



ในตัวอย่างนี้จะแสดงให้เห็นว่าทำไมบางภาพถึงควรทำ Thresholding ก่อนการทำ Canny Edge Detection ในตัวอย่างทางซ้ายจะเห็นว่ากระเป๋ามีลวดลาย Texture ที่ละเอียดทำให้การหา Edge นั้นทำได้ยากและผลลัพธ์ของ Edge ที่ได้นั้นส่วนมากเป็น Noise ที่ไม่ได้ต้องการ แต่ตัวอย่างทางขวานั้นหาเราทำ Thresholding ก็จะทำให้สูญเสีย Edge ในส่วนที่เป็น Logo ไป ดังนั้นกระบวนการ Process ภาพในครั้งนี้ควรลำดับและเลือก Preprocess ต่างๆให้เหมาะสมในแต่ละภาพเพื่อให้ได้ผลลัพธ์ที่ดี

ตัวอย่าง Code

```
edges = cv.Canny(blur, 0, 170, apertureSize=3, L2gradient=True)
```

5. Morphological Transformations

ในรายวิชานี้ไม่ได้มีเนื้อหาเกี่ยวกับการทำ Image Morphological แต่เนื่องจากได้ลองทำ Operation ที่เกี่ยวข้องกับด้านนี้แล้วทำให้ผลลัพธ์การทำ Logo Segmentation นั้นดีขึ้น โดยอยู่บนสมมติฐานที่ว่า Shape ของ Logo บนกระเป๋านั้นเป็น Closed Shaped ที่เรารู้จักอยู่แล้วและการทำ Morphological Operation กับ Logo ที่มี Shape นั้นจะทำให้เราสามารถ Extract Logo ออกจากพื้นหลังและกระเป๋าได้ดีขึ้น โดยมีการใช้งาน Morphological Transformations 3 แบบดังนี้

5.1. Dilation

ก่อนจะทำความเข้าใจเกี่ยวกับการใช้ Dilation มาทำความรู้จักกับ Erosion กันก่อน Erosion มาจาก Erode ที่แปลว่ากัดกร่อน โดย Method Erosion มี Input 2 พื้นฐานอย่างคือรูปเดิมโดยมาค่าของรูปเป็นแบบ Binary (ขาวและดำ) และ Kernel โดย Erosion จำเอา Kernel เลื่อนไปตามรูปภาพโดยถ้า Pixel ทั้งหมดใต้ Kernel มีค่าเป็น 1 ก็จะรักษาค่าเดิมที่เป็น 1 ของรูปภาพไว้ และถ้าหากมี Pixel ใด Pixel หนึ่งภายใต้ Kernel ไม่ได้มีค่าเท่ากับ 1 ก็จะกร่อน (Erode) Pixel นั้นทิ้ง (เปลี่ยนจาก 1 เป็น 0)

โดย Dilation ก็จะทำงานตรงกันข้ามกับ Erosion โดย ถ้าหากมี Pixel ใด Pixel หนึ่งภายใต้ Kernel ก็จะเปลี่ยน Pixel ตรงกลาง Kernel เป็น 1 ทำให้ “ขอบ” ของภาพขยายออก

ตัวอย่างผลลัพธ์

https://docs.opencv.org/3.4/dilation.png

ตัวอย่าง Code

```
dilation = cv.dilate(img,kernel,iterations = 1)
```

5.2. Closing

Closing เกิดจากการนำ Morphological Operation Erosion และ Dilation มารวมกันโดยจะทำ Erosion ก่อนแล้วจึงทำ Dilation การทำ Closing จะช่วยให้เรากำจัด Noise ที่ไม่ได้อยู่ภายในรูปร่างของเราออก

ตัวอย่างผลลัพธ์

https://docs.opencv.org/3.4/closing.png

ตัวอย่าง Code

```
closing = cv.morphologyEx(img, cv.MORPH_CLOSE, kernel)
```

5.3. Opening

Opening จะทำตรงข้ามกับ Closing โดยนำ Morphological Operation Erosion และ Dilation มารวมกันโดยจะทำ Dilation ก่อนแล้วจึงทำ Erosion การทำ Opening จะช่วยให้เรากำจัด Noise ที่ภายในรูปร่างของเราออกโดยจะช่วยเติมช่องว่างภายในรูปภาพ

ตัวอย่างผลลัพธ์

https://docs.opencv.org/3.4/opening.png

ตัวอย่าง Code

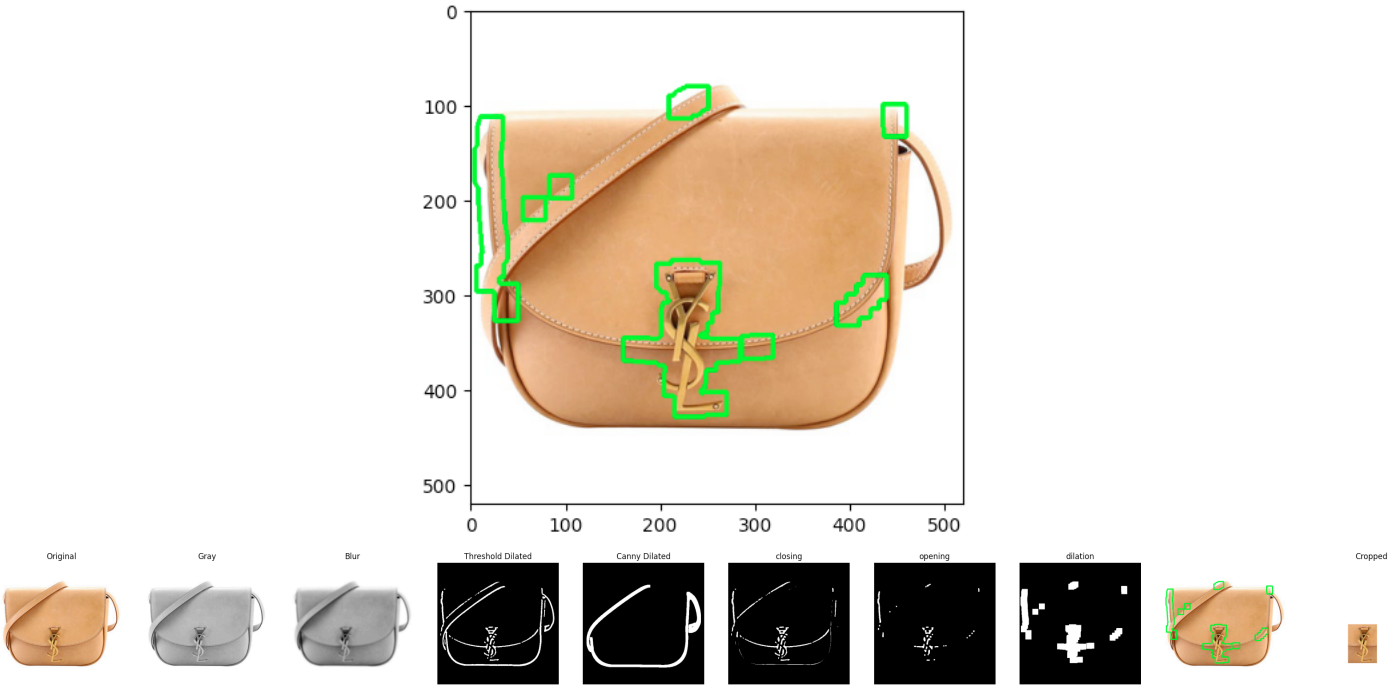
```
opening = cv.morphologyEx(img, cv.MORPH_OPEN, kernel)
```

6. Image Contours

การทำ Image Contours ก็เป็นอีกหนึ่งหัวข้อที่รายวิชานี้ไม่ได้มีเนื้อหาเกี่ยวข้อง แต่ได้นำมาใช้ในการทำ Logo Segmentation ในครั้งนี้ สาเหตุที่ได้นำ Image Contour มาใช้ในครั้งนี้คือในตอนแรกที่พยายามตัดส่วนที่เป็น Logo ออกมา พบว่าจะมี Noise เล็กส่วนที่ไม่ใช่ Logo นั้นติดมาด้วย จึงใช้ Image Contour ในการหา Boundary และ Detect ส่วนที่เป็น Logo โดยหาพื้นที่ Contour ที่ใหญ่ที่สุด

แล้ว Contour คืออะไร Contour คือการลากเส้นเชื่อมจุดที่อยู่ใกล้กันตามแนวขอบโดยจุดที่ Contour เลือกที่จะเชื่อมนั้นต้องมี Intensity เท่ากัน การใช้งาน Contour เป็นประโยชน์เป็นอย่างมากในการทำ Shape Analysis และ Object Detection ซึ่งนั่นก็เป็นสิ่งที่เราพยายามทำอยู่

ตัวอย่างผลลัพธ์



ในตัวอย่างข้างต้นผ่านการทำ Operation ที่นำเสนอไปข้างต้นแล้วนำผลลัพธ์ที่ได้ไปทำ Contour ต่อผลลัพธ์ของ Contour ที่ได้จะถูก Highlight ด้วยสีเขียวนั้นคือ Contour ทั้งหมดใน Dilation โดยมี Logo อยู่ภายใต้ Contour ที่มีขนาดใหญ่ที่สุด

ตัวอย่าง Code

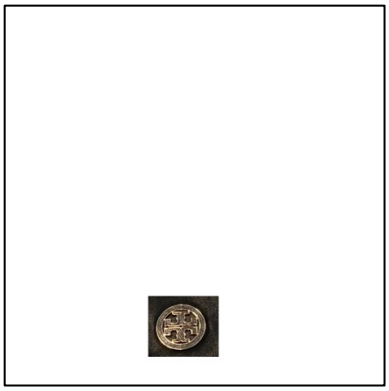
```
drawContours = cv.drawContours(img.copy(), contours, -1, (0, 255, 0), 3)
```

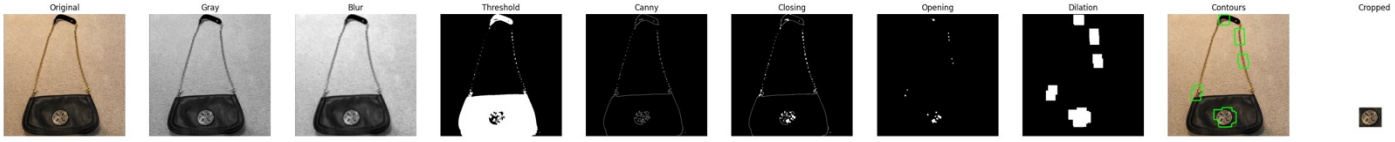
เมื่อนำทุก Operation ที่ได้นำเสนอไปมาทำงานร่วมกันจะทำให้สามารถทำ Logo Segmentation ได้อย่างมีประสิทธิภาพ โดยในแต่ละรูปภาพอาจจะต้องมีการ Tuning Parameter และค่าที่เหมาะสมสำหรับในแต่ละรูปอาจจะมีค่าแตกต่างกันไป ใน Section ถัดไปจะพูดถึงผลลัพธ์และวิเคราะห์ผลลัพธ์ที่ได้

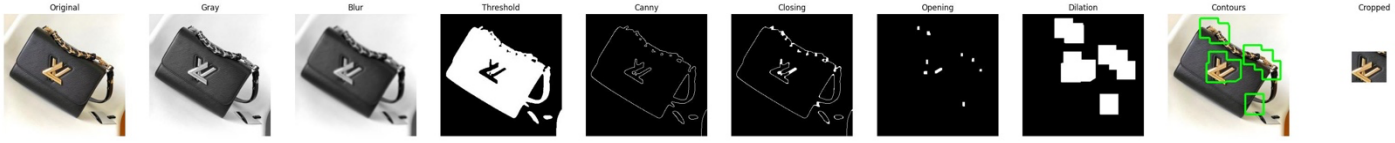
Result and Discussion

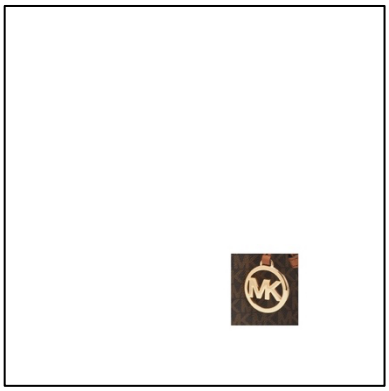
ใน Section นี้จะแสดงถึงผลลัพธ์โดยจะนำเสนอผลลัพธ์ที่ได้ในแต่ละขั้นตอนของ Operation ต่าง ๆ เพื่อนำมาวิเคราะห์และลอง Tuning Parameter ให้มีความเหมาะสมกับภาพนั้น ๆ

สำหรับ Segmentation นั้นได้มีการตัดเป็น Rectangle Boundary แทนที่จะตัดตามรูปร่างของ Logo เนื่องจากได้ลองตัดตามรูปร่างของ Logo แล้วพบว่าการตัดตามส่วนของรูปภาพนั้นเป็นไปได้ค่อนข้างยากจึงได้ลองตัดตาม Bounding Box ของ Logo แทน ผลลัพธ์คือ

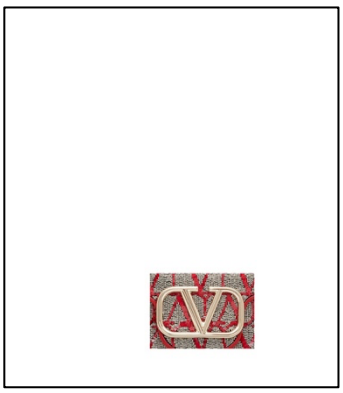


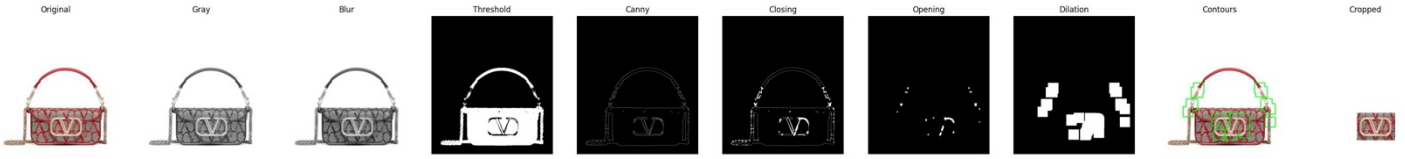






ใน 3 ภาพแรกนี้ได้นำมาทำด้วยค่า Parameter ต่างๆที่เท่ากัน จากการได้สังเกตภาพจะพบเห็นว่าทั้ง 3 ภาพนี้มีความเหมือนคือ พื้นหลังกับบริเวณของ Logo นั้นมีความ Contrast กันโดยมีสีกระเป๋าเป็นสีดำหรือน้ำตาลเข้มและมีสี Logo เป็นสีทอง แต่ก็ยังมีความท้าทายคือบริเวณที่ของสายของกระเป๋าในภาพแรกและภาพที่ 2 นั้นมีสีที่ใกล้เคียงกับ Logo หากสังเกต Contour ของรูปภาพจะเห็นว่ามี Contour บางส่วนอ่อนไหวต่อบริเวณที่เป็นสายด้วย แต่ถ้าพิจารณาถึงขนาดพื้นที่ของ Contour แล้วจะเห็นว่าบริเวณที่เป็น Logo นั้นจะใหญ่กว่ามาก





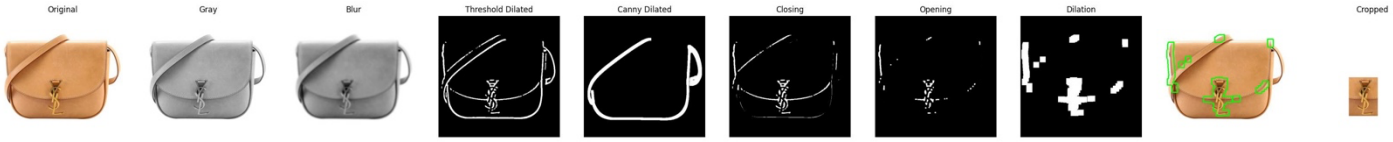
ในภาพนี้ความยากคือกระเป๋ามีลวดลายและพื้นผิวที่ไม่สม่ำเสมอถ้าหากนำรูปที่ผ่านการทำ Gaussian Blur ไปทำ Canny Edge Detection และจะพบว่าจะได้ Edge ตามลวดลายของกระเป๋าด้วย โดยในภาพนี้ได้มีการปรับ Parameter 2 ส่วน ในส่วนแรกคือปรับ lower bound และ upper bound ของ Canny เพื่อให้ Edge ที่ได้ยังครบและมีการขยายขนาดของ Kernel ในการทำ Morphological Operation เป็น 7x7 เพื่อลบ Noise ออก

ในภาพนี้ความยากคือ Logo และกระเป๋าเป็นสีเดียวกันแล้วขอบของ Logo นั้นมีสีที่ไม่ค่อยแตกต่างกับตัวกระเป๋ามากนัก และตัวกระเป๋ามีพื้นผิวที่ขรุขระ ทำให้ Canny ตรวจพื้นผิวของกระเป๋าเป็น Edge ด้วย โดยในภาพนี้มีทั้งการเปลี่ยน Method ของการทำ Thresholding เป็น Adaptive Gaussian Thresholding และมีการปรับ Parameter 3 ส่วนคือปรับขนาดของ Gaussian Blur เป็นขนาด 15x15 เพื่อลด Texture กระเป๋า ต่อมาลดขนาดของ Kernel ในการทำ Morphological Operation เป็น 3x3 เพื่อให้ยังคงไว้ซึ่งรายละเอียดของ Logo บนกระเป๋า และปรับ lower bound และ upper bound ของ Canny สุดท้ายมีการเพิ่มเติม operation ในการลบ Edge ที่ไม่ต้องการออกโดยนำผลลัพธ์ที่ได้จากการทำ Threshold ลบกับผลลัพธ์ที่ได้จากการทำ Canny Edge Detection โดย Canny Edge Detection นั้นจะได้เฉพาะ Edge ขอบของกระเป๋าเมื่อนำมาลบกับ Threshold ก็จะเหลือเพียง Edge บริเวณ Logo[2]

Code

```
edges_dil = cv.dilate(edges, np.ones((3, 3), np.uint8), iterations=30)
logo_edge = thr - edges_dil
```

[2] Edge ของกระเป๋าได้มีการทำ Dilation 30 ครั้งเพื่อให้เส้นของ Edge มีขนาดใหญ่ขึ้นโดยเมื่อนำมา Threshold ลบด้วย Edges แล้วจะทำให้ Edge ของกระเป๋าหายไปหมด



ภาพนี้มีความคล้ายคลึงกันกับภาพด้านก่อน มีการเพิ่มเติมในการทำ Morphological Operation กับ Threshold และปรับ lower bound และ upper bound ของ Canny

Morphological Operation กับ Threshold

```
thr = cv.adaptiveThreshold(blur, 255, cv.ADAPTIVE_THRESH_GAUSSIAN_C, cv.THRESH_BINARY_INV, 11, 2)
kernel = np.ones((3, 3), np.uint8)
closing_thr = cv.morphologyEx(thr, cv.MORPH_CLOSE, kernel)
opening_thr = cv.morphologyEx(closing_thr, cv.MORPH_OPEN, kernel)
```