Báo cáo về bài tập giữa kì môn Xử lý ảnh (lớp INT3404_1)

Họ và tên: Đỗ Đức Huy

Mã sinh viên: 21020124

I. Task 1: Làm data cho game "Spot the differences"

```
[ ] import matplotlib.pyplot as plt
    import cv2
    import numpy as np
    import random
    import glob
    import os
```

```
[ ] drive.mount('./drive')
```

```
Mounted at ./drive
```

```
[ ] pwd
```

```
'/content'
```

```
cd /content/drive/MyDrive/DigitalImageProcessingMidterm
```

```
/content/drive/MyDrive/DigitalImageProcessingMidterm
```

1. Định nghĩa một số hàm tiện ích cần dùng

```
[ ] # Hàm dùng để resize lại ảnh
    def resizeImage(image, height):
        h, w, _ = image.shape
        ratio = w / h
        width = int(ratio * height)
        image = cv2.resize(image, (width, height), interpolation= cv2.INTER_CUBIC)
        return image
```

```
# Hàm lấy độ dài đường chéo của ảnh
def getDiameter(img):
    h, w = img.shape[:2]
    diameter = int(np.sqrt(h ** 2 + w ** 2)) + 1
    return diameter
```

```
# Hàm thêm border vào ảnh
def addBorders(img):
    h, w = img.shape[:2]
    diameter = getDiameter(img)
    top = bottom = diameter - h
    left = right = diameter - w
    img_border = cv2.copyMakeBorder(
        img, top, bottom, left, right, borderType=cv2.BORDER_CONSTANT, value = (0, 0, 0)
    )
    return img_border
```

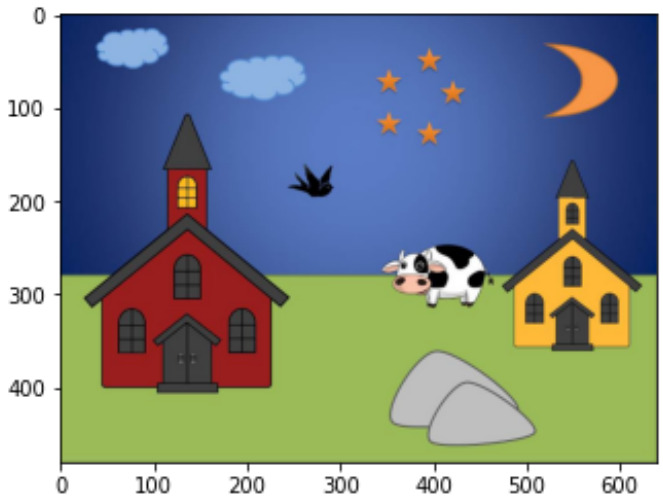
```
# Hàm xóa border của ảnh
def removeBorders(img):
    h, w = np.shape(img)[:2]

    B = cv2.cvtColor(src = img, code = cv2.COLOR_BGR2GRAY)

    left = w
    right = 1
    top = h
    bottom = 1

    for i in range (1, h):
        for j in range (1, w):
            if B[i,j] > 0:

                if i < top:
                    top = i

                if i > bottom:
                    bottom = i

                if j < left:
                    left = j

                if j > right:
                    right = j

    C = img[top: bottom + 1, left: right + 1]

    return C
```

```
[ ]
        return C
```

```
[ ]  # Hàm lưu trữ ảnh kết quả
     def saveImage(output_dir, filename, image):
         if not os.path.exists(output_dir):
             os.makedirs(output_dir)
         return cv2.imwrite(os.path.join(output_dir, filename), image)
```

```
[ ]  # Load ảnh cần làm data
     filename = 'church0.jpg'
     image = cv2.imread(os.path.join('./images', filename))

     image = resizeImage(image, 480)
     areaImage = image.shape[0] * image.shape[1]

     plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f716cfb6670>
```



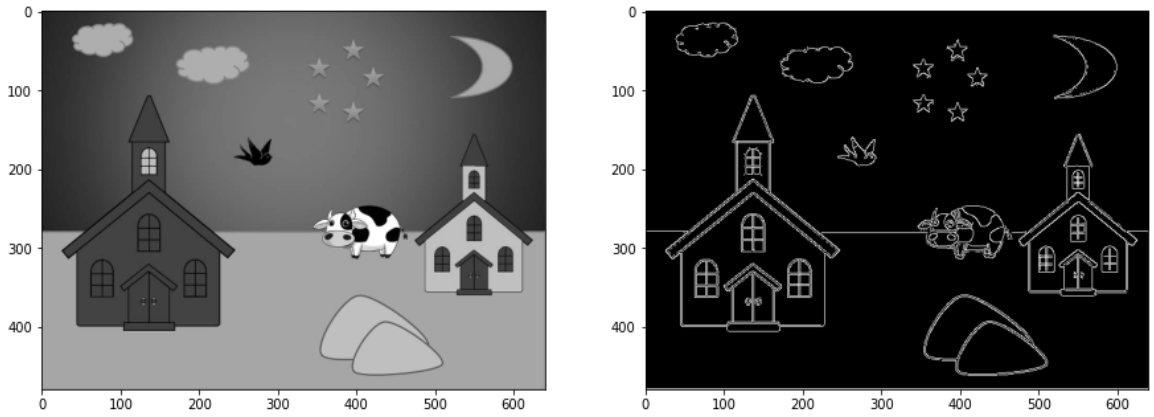

```
[ ]  imageClone = image.copy()
```

+ Code

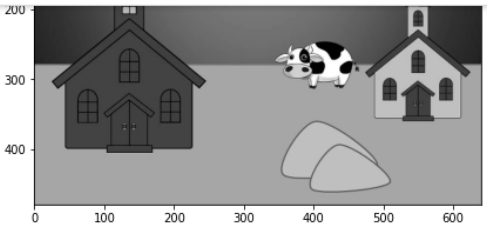
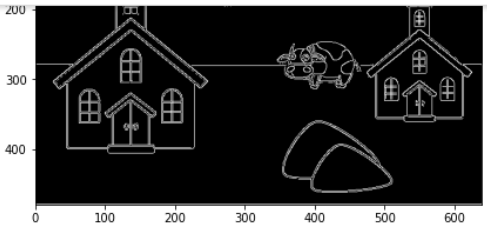
```
[ ]  imageClone = image.copy()
```

```
[ ]  # Tìm cạnh của ảnh
     plt.figure(figsize = (15, 15))
     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
     edge = cv2.Canny(gray, 30, 180)
     plt.subplot(1, 2, 1)
     plt.imshow(gray, cmap = 'gray')
     plt.subplot(1, 2, 2)
     plt.imshow(edge, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7f7154f13fd0>
```

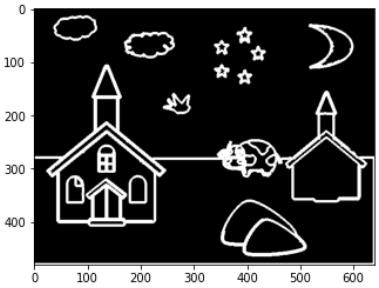



```
[ ]  # Tìm contour của ảnh
     contours, _ = cv2.findContours(edge, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
     mask = np.zeros(shape = image.shape)
```

```
# Tìm contour của ảnh
contours, _ = cv2.findContours(edge, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
mask = np.zeros(shape = image.shape)
cv2.drawContours(mask, contours, -1, (255, 255, 255), 3)
plt.imshow(mask, cmap = 'gray')
```

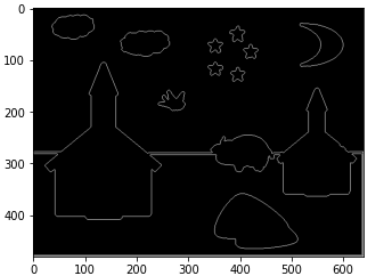
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
<matplotlib.image.AxesImage at 0x7f7169566580>
```



```
# Vì các contour chi tiết rất dễ gây nhiễu cho việc thực hiện các thuật toán nên
# ta một lần nữa thực hiện tìm contour của ảnh nhưng ở mức tổng quát hơn
mask = mask.astype('uint8')
mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
```

```
# Vì các contour chi tiết rất dễ gây nhiễu cho việc thực hiện các thuật toán nên
# ta một lần nữa thực hiện tìm contour của ảnh nhưng ở mức tổng quát hơn
mask = mask.astype('uint8')
mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
mask2 = np.zeros(shape = image.shape)
contours2, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(mask2, contours2, -1, (255, 255, 255), 1)
plt.imshow(mask2, cmap = 'gray')
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
<matplotlib.image.AxesImage at 0x7f71695d7fd0>
```



```
# Lựa chọn các contour của các vật thể để thực hiện các phép biến đổi
contour_test = []
for i in range(len(contours2)):
    contourArea = cv2.contourArea(contours2[i])
    if contourArea > 0.0025 * areaImage and contourArea < areaImage * 0.2:
        contour_test.append(contours2[i])
contour_test = sorted(contour_test, key = cv2.contourArea)
numOfContour = len(contour_test)
```

2. Các cách thay đổi ảnh để tạo ra data cho trò chơi

a. Thêm vật vào ảnh

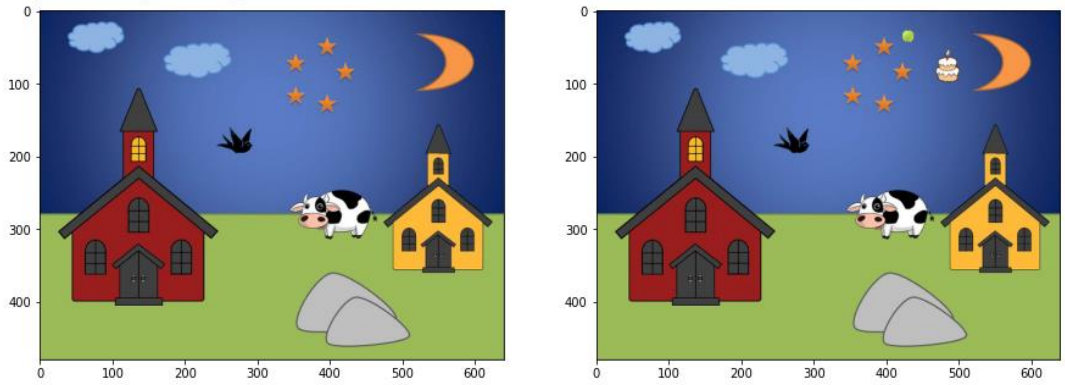
```python
def addObject(image, filepath, contours, amount):
    # Tạo ra mask
    mask = np.zeros(shape = image.shape)

    # Lấy tất cả ảnh vật có thể thêm vào
    pngfile = glob.glob(os.path.join(filepath, '*.png'))
    if amount < 0:
        amount = random.randint(1, len(pngfile))

    # Tính vị trí của tất cả contour
    area = []
    for i in range(len(contours)):
        (x, y, w, h) = cv2.boundingRect(contours[i])
        area.append([x, y, w, h])

    # Thực hiện thêm vật vào ảnh
    for count in range(amount):

        # Chọn vật ngẫu nhiên trong tất cả ảnh có sẵn
        index = random.randint(0, len(pngfile) - 1)
        add_image = cv2.imread(pngfile[index])
        h, w, _ = add_image.shape

        # Chọn vị trí ngẫu nhiên để vật xuất hiện trong ảnh
        n = 0
        while (n < 20):
            x = random.randint(0, image.shape[1] - 1)
            y = random.randint(0, image.shape[0] - 1)
```

```
    while (n < 20):
        x = random.randint(0, image.shape[1] - 1)
        y = random.randint(0, image.shape[0] - 1)

        #Kiểm tra vị trí có hợp lệ không (không đè vào contour của các vật có trong ảnh)
        check = True
        for i in range(len(area)):
            if area[i][0] < x and area[i][0] + area[i][2] > x and area[i][1] < y and area[i][1] + area[i][3] > y:
                check = False
                break
            if area[i][0] < x + w and area[i][0] + area[i][2] > x + w and area[i][1] < y + h and area[i][1] + area[i][3] > y + h:
                check = False
                break

            if (x + w) > image.shape[1]:
                check = False
                break
            if (y + h) > image.shape[0]:
                check = False
                break

        # Nếu vị trí hợp lệ thì thực hiện thêm vật vào ảnh
        if check:

            # Tạo mask chứa vật thể cần thêm vào
            mask = cv2.cvtColor(add_image, cv2.COLOR_BGR2GRAY)
            _, mask = cv2.threshold(add_image, 0, 255, cv2.THRESH_BINARY)
            mask = 255 - mask

            #Lấy vùng ảnh cần thay thế bằng vật
            local = image[y: y + h, x: x + w]

            # Thực hiện toán tử bitwise_and và bitwise_or để gán vật vào ảnh
            local = cv2.bitwise_and(local, mask)
            local = cv2.bitwise_or(local, add_image)
            image[y: y + h, x: x + w] = local
            area.append([x, y, w, h])
            break
        else:
            n += 1
return image
```

```
    return image
```

```
imageClone1 = image.copy()
# Thêm 2 vật bất kì vào ảnh
imageClone1 = addObject(imageClone1, './add_images', contour_test, 2)

plt.figure(figsize = (15, 15))
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(imageClone1, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f7154c8c8b0>
```



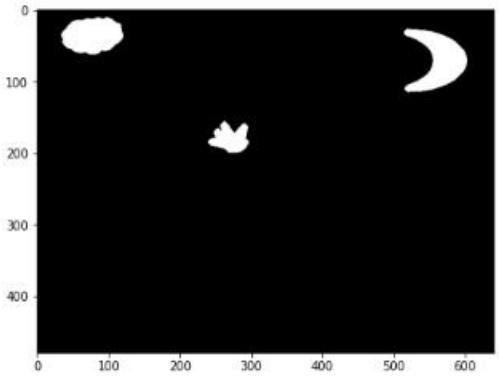
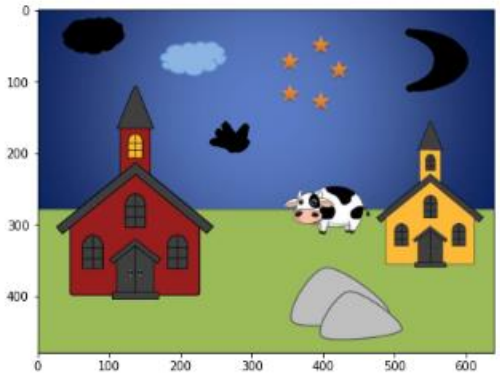
b. Xóa vật trong ảnh

b. Xóa vật trong ảnh

```python
def deleteObject(image, contours):
    image = image.copy()
    maskList = []
    mask = np.zeros(shape = image.shape)
    for i in range(len(contours)):

        # Lấy contour của các vật thể cần xóa và làm mask từ các contour đó
        cv2.drawContours(mask, contours, i, (255, 255, 255), 3)
        cv2.fillPoly(mask, [contours[i]], color = (255, 255, 255))
        (x, y, w, h) = cv2.boundingRect(contours[i])
        maskList.append([x, y, w, h])

    # Sau khi tìm được mask sẽ thực hiện toán tử bitwise_and để thức hiện xóa vật thể trong ảnh
    plt.figure(figsize = (15, 15))
    mask = mask.astype('uint8')
    plt.subplot(1, 2, 1)
    plt.imshow(cv2.cvtColor(mask, cv2.COLOR_BGR2RGB))

    mask1 = 255 - mask
    image = cv2.bitwise_and(image, mask1)
    plt.subplot(1, 2, 2)
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

    # Sau khi xóa vật thể sẽ xuất hiện các lỗ trống trong ảnh
    # Sử dụng cv2.inpaint để khôi phục lại các lỗ trống đó
    mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
    image = cv2.inpaint(image, mask, 3, cv2.INPAINT_TELEA)

    # Sau khi khôi phục lỗ trống, sẽ xuất hiện nhiều nhiễu ở vùng lỗ trống
    # Sử dụng hàm denoise của opencv để khử nhiễu(có thể sử dụng GaussianBlur thay thế)
    for i in range(len(maskList)):
        x, y, w, h = maskList[i]
        local = image[y: y + h, x: x + w]
        local = cv2.fastNlMeansDenoisingColored(local, None, 10, 10, 7, 21)
        image[y: y + h, x: x + w] = local
    return image
```

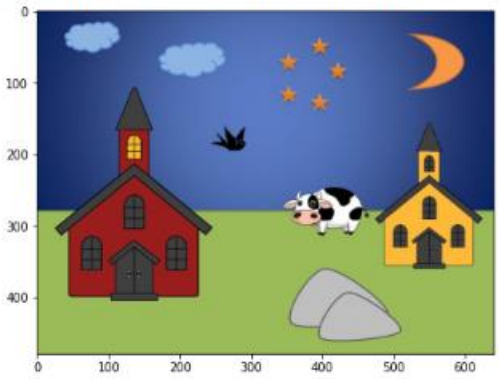
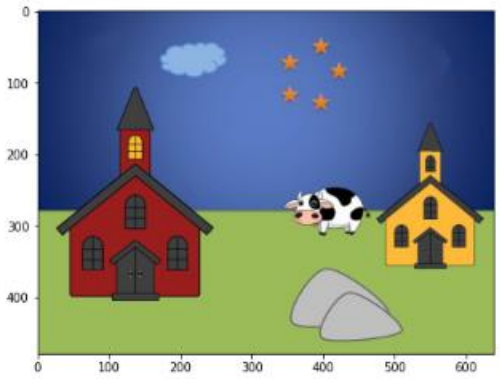
```
        return image
```

```
[ ]  imageClone2 = image.copy()
     # Xóa các vật đã được chọn trong contour
     imageClone2 = deleteObject(imageClone2, contour_test[:3])
```

```
[ ]  plt.figure(figsize = (15, 15))
     plt.subplot(1, 2, 1)
     plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
     plt.subplot(1, 2, 2)
     plt.imshow(cv2.cvtColor(imageClone2, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f715519a100>
```

c. Xoay vật trong ảnh

c. Xoay vật trong ảnh

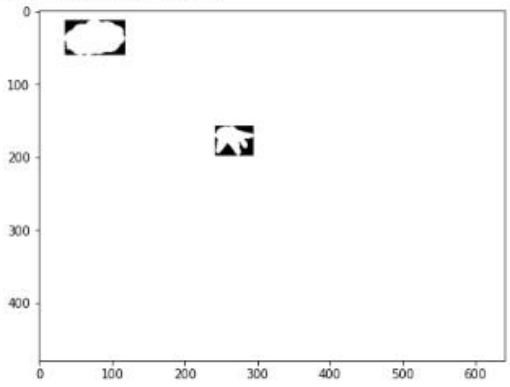
```python
def rotateObject(image, contours):
    image = image.copy()
    mask = np.zeros(shape = image.shape)

    for i in range(len(contours)):
        (x, y, w, h) = cv2.boundingRect(contours[i])
        if not (x == 0 or x + w == image.shape[1] or y == 0 or y + h == image.shape[0]):

            # Tạo ra mask của các vật thể trong ảnh cần xoay
            cv2.drawContours(mask, contours, i, (255, 255, 255), 1)
            cv2.fillPoly(mask, [contours[i]], color = (255, 255, 255))

            # Chọn vùng ảnh cần xoay
            # Cần phải thêm border vào vùng ảnh vì khi xoay có thể dẫn đến bị mất
            # thông tin ảnh ở viền ảnh
            local = image[y: y + h, x: x + w]
            local = addBorders(local)

            # Chọn vùng mask tương ứng cần xoay theo
            # Cũng cần phải thêm border
            mask_local = mask[y: y + h, x: x + w]
            mask_local = 255 - mask_local
            mask_local = addBorders(mask_local)

            angle = 180
            M = cv2.getRotationMatrix2D((local.shape[1] // 2, local.shape[0] // 2), angle, 1)

            # Thực hiện xoay ở vùng ảnh đã chọn
            # Sau khi xoay xong, phải xóa bỏ phần border đã thêm vào
            fake = cv2.warpAffine(local, M, (local.shape[1], local.shape[0]))
            fake = removeBorders(fake)
            image[y: y + h, x: x + w] = fake

            # Thực hiện xoay ở vùng mask tương ứng
            # Sau khi xoay xong, cũng xóa phần border
```

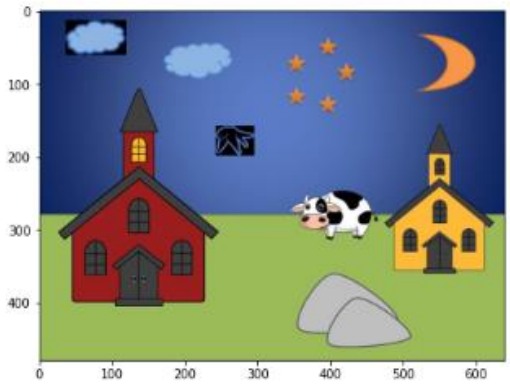
```
                # Thực hiện xoay ở vùng ảnh đã chọn
                # Sau khi xoay xong, phải xóa bỏ phần border đã thêm vào
                fake = cv2.warpAffine(local, M, (local.shape[1], local.shape[0]))
                fake = removeBorders(fake)
                image[y: y + h, x: x + w] = fake

                # Thực hiện xoay ở vùng mask tương ứng
                # Sau khi xoay xong, cũng xóa phần border
                fake_mask = cv2.warpAffine(mask_local, M, (mask_local.shape[1], mask_local.shape[0]))
                fake_mask = removeBorders(fake_mask.astype('uint8'))
                mask[y: y + h, x: x + w] = fake_mask

        # Sử dụng toán tử bitwise_and để xóa phần bên ngoài vật thể được xoay
        mask = mask.astype('uint8')
        mask = 255 - mask
        image = cv2.bitwise_and(image, mask)

        plt.figure(figsize = (15, 15))
        plt.subplot(1, 2, 1)
        plt.imshow(cv2.cvtColor(mask, cv2.COLOR_BGR2RGB))
        plt.subplot(1, 2, 2)
        plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))

        # Sau khi thực hiện xóa phần bên ngoài vật thể, sẽ có các lỗ trống trong ảnh
        # Sử dụng cv2.inpaint để thực hiện khôi phục các lỗ trống
        mask = 255 - mask
        mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
        print(mask.shape, image.shape)
        image = cv2.inpaint(image, mask, 3, cv2.INPAINT_TELEA)

        return image
```

```
[ ]  imageClone3 = image.copy()
```

```
[ ] imageClone3 = image.copy()

    # Thực hiện xoay vật đã chọn trong contour
    imageClone3 = rotateObject(imageClone3, contour_test[:2])
```

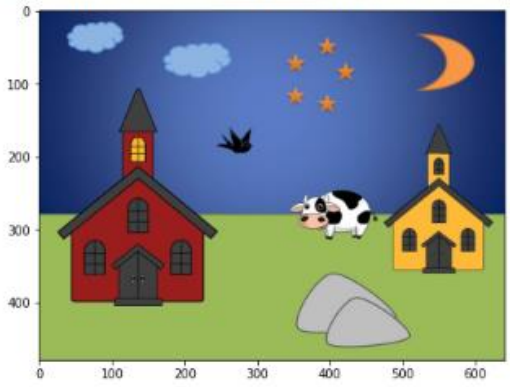
```
(480, 640) (480, 640, 3)
```





```
plt.figure(figsize = (15, 15))
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(imageClone3, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f71552a1cd0>
```



▸ d. Thay đổi màu của vật trong ảnh

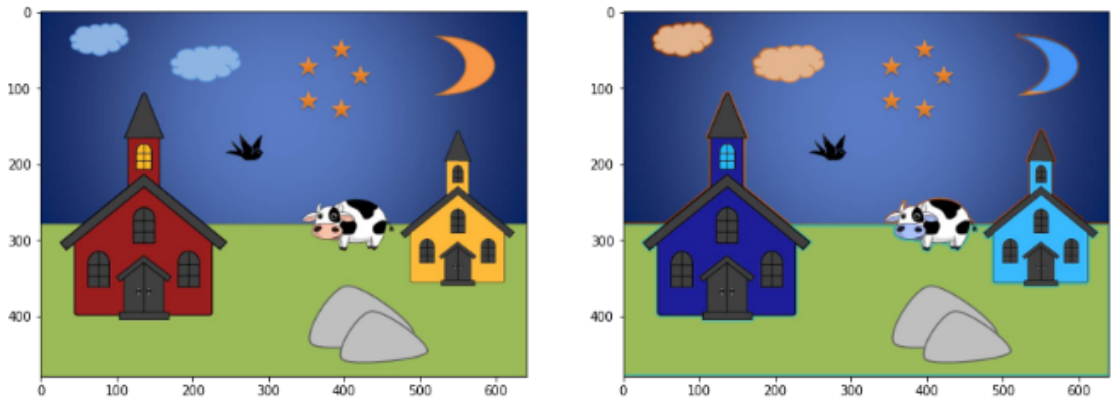
d. Thay đổi màu của vật trong ảnh

```
[ ]  def changeColorObject(image, contours):
         image = image.copy()
         mask = np.zeros(shape = image.shape)
         for i in range(len(contours)):

             # Lấy contour của các vật thể cần xóa và làm mask từ các contour đó
             cv2.drawContours(mask, contours, i, (255, 255, 255), 1)
             cv2.fillPoly(mask, [contours[i]], color = (255, 255, 255))
             (x, y, w, h) = cv2.boundingRect(contours[i])

             # Thực hiện thay đổi màu của vật đã chọn
             # Thay đổi màu vật dựa trên việc thay đổi màu của từng pixel trong ảnh
             for j in range(y, y + h):
                 for k in range(x, x + w):
                     if mask[j, k, 0] == 255 and mask[j, k, 1] == 255 and mask[j, k, 2] == 255:
                         image[j, k, :] = image[j, k, ::-1]

         return image
```

```
[ ]  imageClone4 = image.copy()

     # Thực hiện thay đổi màu của vật đã được chọn trong contour
     imageClone4 = changeColorObject(imageClone4, contour_test)
```

```
[ ]  plt.figure(figsize = (15, 15))
     plt.subplot(1, 2, 1)
     plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
     plt.subplot(1, 2, 2)
     plt.imshow(cv2.cvtColor(imageClone4, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f7155159700>
```

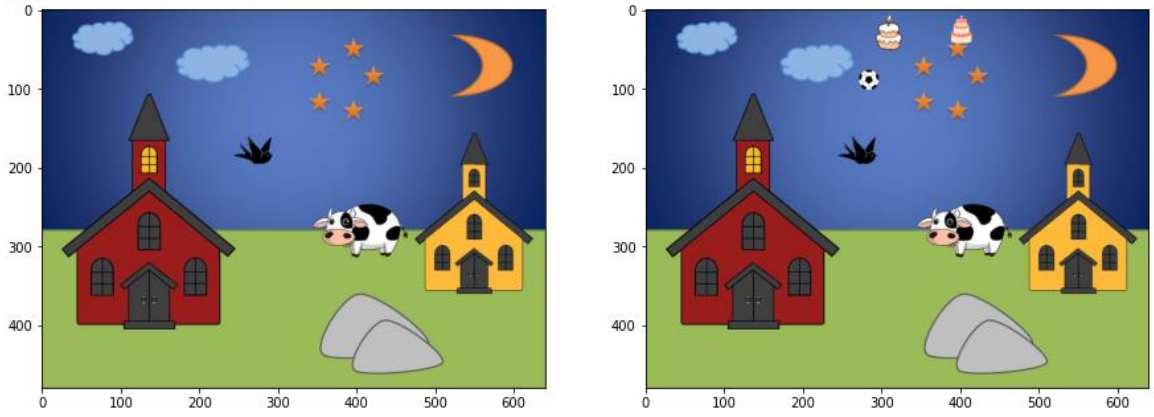


3. Định nghĩa các cấp độ của trò chơi

Level 1: Sinh data bằng cách thêm vật vào ảnh

```
[ ]  level1 = image.copy()
     amount = 3 # Số lượng vật cần thêm vào (amount < 0 <=> thêm vật với số lượng bất kì)
     level1 = addObject(level1, './add_images', contour_test, amount)

     plt.figure(figsize = (15, 15))
     plt.subplot(1, 2, 1)
     plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
     plt.subplot(1, 2, 2)
     plt.imshow(cv2.cvtColor(level1, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f7154a0f730>
```




```
[ ]  saveImage('./game_data', 'level1.jpg', level1)
```

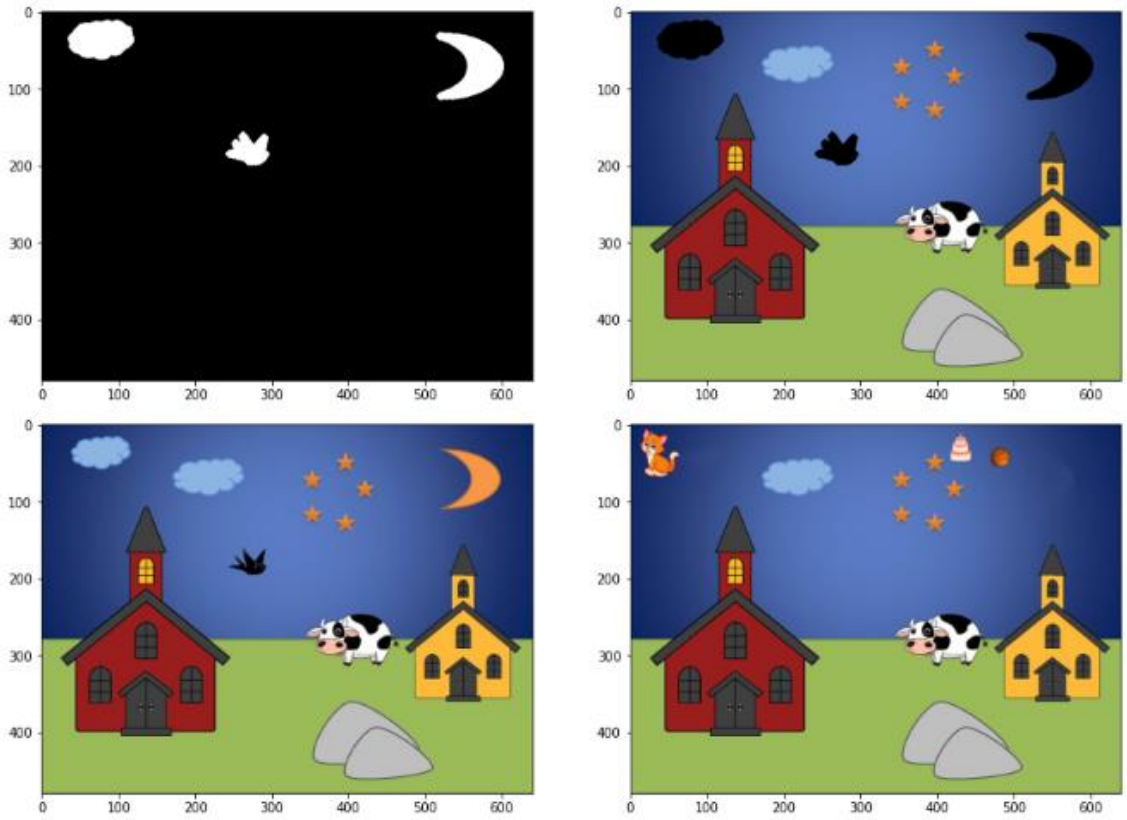
```
True
```

Level 2: Sinh data bằng cách

- Thêm vật vào ảnh
- Xóa vật trong ảnh

```
[ ]  level2 = image.copy()

     amount = 3
     level2 = deleteObject(level2, contour_test[:int(numOfContour / 2)])
     level2 = addObject(level2, './add_images', contour_test, amount)

     plt.figure(figsize = (15, 15))
     plt.subplot(1, 2, 1)
     plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
     plt.subplot(1, 2, 2)
     plt.imshow(cv2.cvtColor(level2, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f715493e190>
```



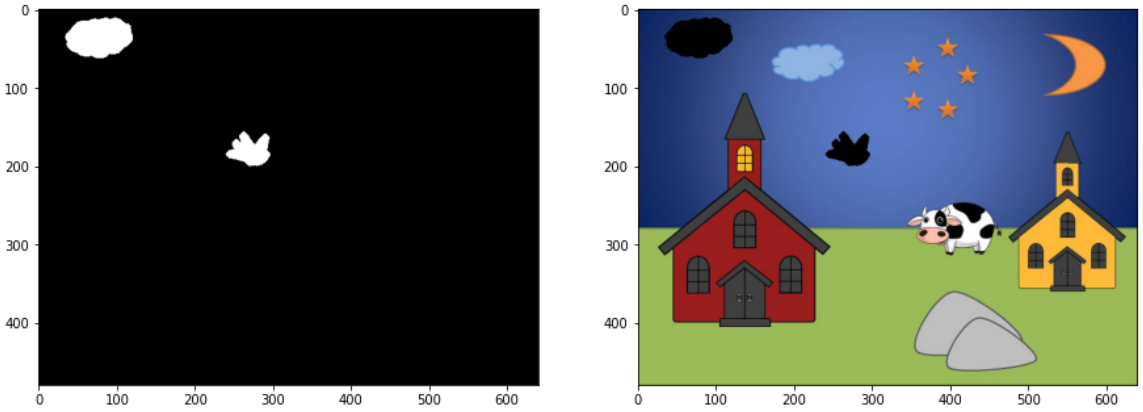
```
[ ]  saveImage('./game_data', 'level2.jpg', level2)
```

Level 3: Sinh data bằng cách

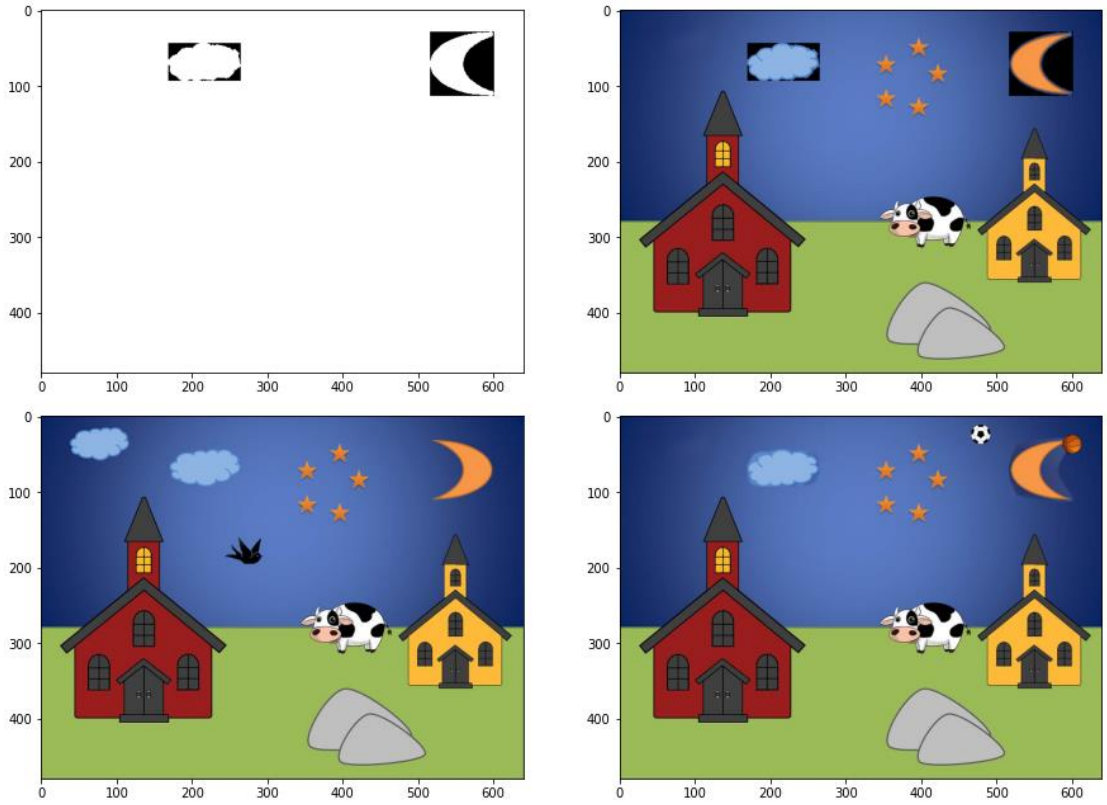
- Thêm vật vào ảnh
- Xóa vật trong ảnh
- Xoay vật trong ảnh

```
level3 = image.copy()

amount = 3
level3 = deleteObject(level3, contour_test[:int(numOfContour / 3)])
level3 = rotateObject(level3, contour_test[int(numOfContour / 3): (2 * int(numOfContour /  3))])
level3 = addObject(level3, './add_images', contour_test, amount)

plt.figure(figsize = (15, 15))
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(level3, cv2.COLOR_BGR2RGB))
```

```
(480, 640) (480, 640, 3)
<matplotlib.image.AxesImage at 0x7f7154706b80>
```



[ ]



[ ] saveImage('./game_data', 'level3.jpg', level3)

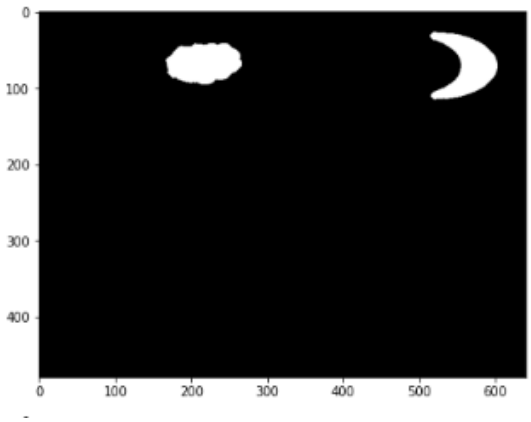
```
True
```

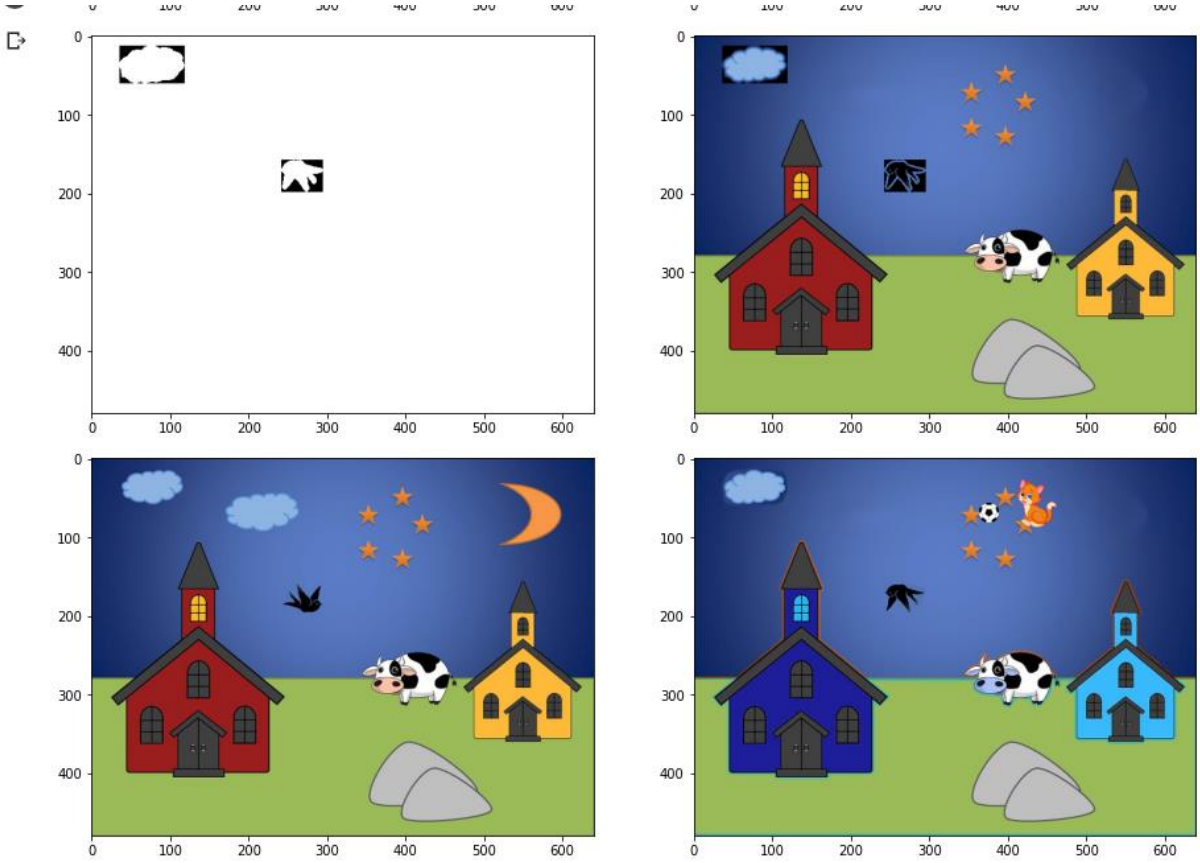
Level 4: Sinh data bằng cách

- Thêm vật vào ảnh
- Xóa vật trong ảnh
- Xoay vật trong ảnh
- Thay đổi màu vật trong ảnh

```
part = int(numOfContour / 3)
level4 = image.copy()

amount = 2
level4 = deleteObject(level4, contour_test[part: (2 * part)])
level4 = rotateObject(level4, contour_test[0: part])
level4 = changeColorObject(level4, contour_test[(2 * part): numOfContour])
level4 = addObject(level4, './add_images', contour_test, amount)

plt.figure(figsize = (15, 15))
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(level4, cv2.COLOR_BGR2RGB))
```

```
(480, 640) (480, 640, 3)
<matplotlib.image.AxesImage at 0x7f7154d21cd0>
```





```
[ ]  saveImage('./game_data', 'level4.jpg', level4)
```

```
True
```

II. Task 2: Làm game “Spot the differences”

```
from google.colab import drive
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt
import cv2
import numpy as np
```

```
[ ] pwd
```

```
'/content'
```

```
[ ] drive.mount('./drive')
```

```
Mounted at ./drive
```

```
[ ] cd /content/drive/MyDrive/DigitalImageProcessingMidterm
```

```
/content/drive/MyDrive/DigitalImageProcessingMidterm
```

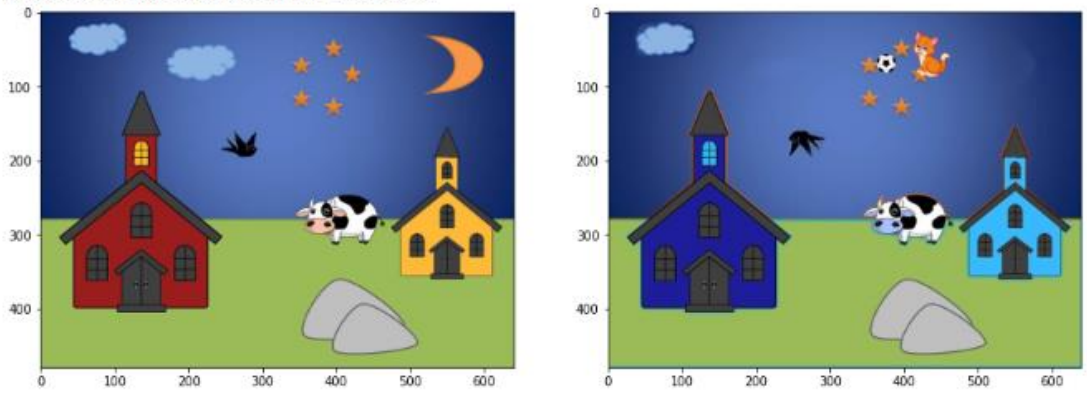
```
[ ] # Hàm resize ảnh
    def resizeImage(image, height):
        h, w, _ = image.shape
        ratio = w / h
        width = int(ratio * height)
        image = cv2.resize(image, (width, height), interpolation= cv2.INTER_CUBIC)
        return image
```

```
[ ] # Load 2 ảnh cần tìm ra điểm khác biệt
    image1 = './images/church0.jpg'
```

```
[ ] # Hàm resize ảnh
    def resizeImage(image, height):
        h, w, _ = image.shape
        ratio = w / h
        width = int(ratio * height)
        image = cv2.resize(image, (width, height), interpolation= cv2.INTER_CUBIC)
        return image
```

```
# Load 2 ảnh cần tìm ra điểm khác biệt
image1 = './images/church0.jpg'
image2 = './game_data/level4.jpg'

image1 = cv2.imread(image1)
image2 = cv2.imread(image2)

# Resize 2 ảnh về cùng kích cỡ để có thể so sánh
# Kích cỡ: width * 480
image1 = resizeImage(image1, 480)
image2 = resizeImage(image2, 480)
print(image1.shape, image2.shape)

plt.figure(figsize = (15, 15))
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image1, cv2.COLOR_BGR2RGB))
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(image2, cv2.COLOR_BGR2RGB))
```

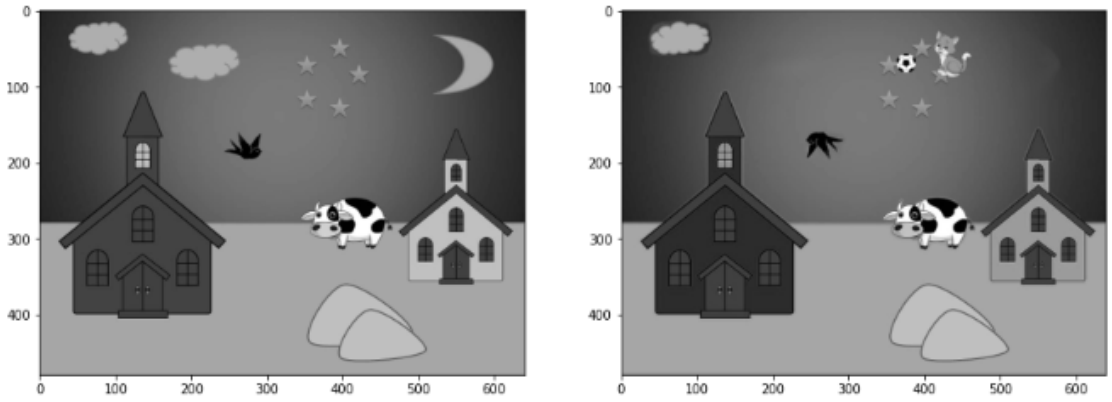
```
(480, 640, 3) (480, 640, 3)
<matplotlib.image.AxesImage at 0x7fbb3c848f10>
```




```
[ ] clone1 = image1.copy()
    clone2 = image2.copy()
```

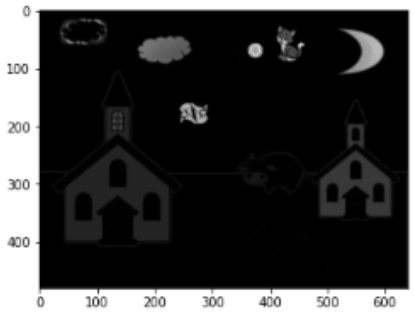
```
[ ]  clone1 = image1.copy()
     clone2 = image2.copy()

     clone1 = cv2.cvtColor(clone1, cv2.COLOR_BGR2GRAY)
     clone2 = cv2.cvtColor(clone2, cv2.COLOR_BGR2GRAY)

     plt.figure(figsize = (15, 15))
     plt.subplot(1, 2, 1)
     plt.imshow(clone1, cmap = 'gray')
     plt.subplot(1, 2, 2)
     plt.imshow(clone2, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7fbb3c5d7d90>
```




```
[ ]  # Tìm điểm khác biệt giữa hai ảnh
     diff = cv2.absdiff(clone1, clone2)
     plt.imshow(diff, cmap = 'gray')
```

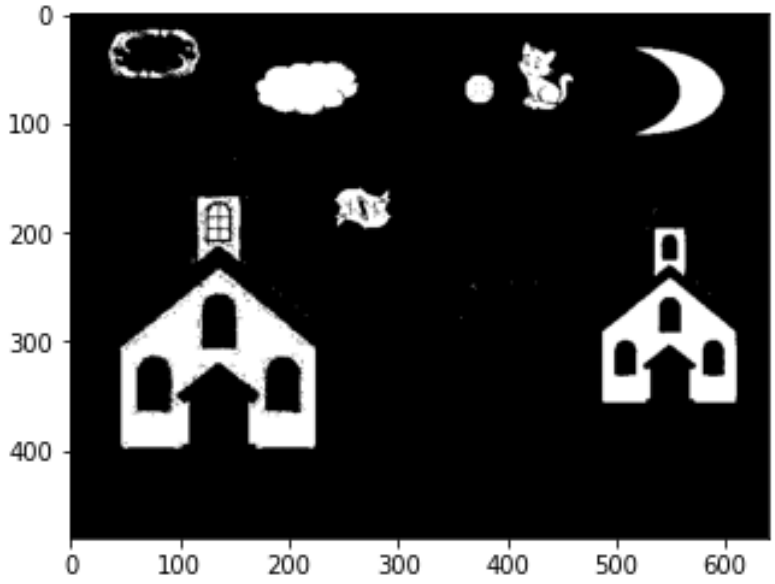
```
<matplotlib.image.AxesImage at 0x7fbb3c35c610>
```




```
[ ]  # Đặt ra threshold để làm nổi bật sự khác nhau
```

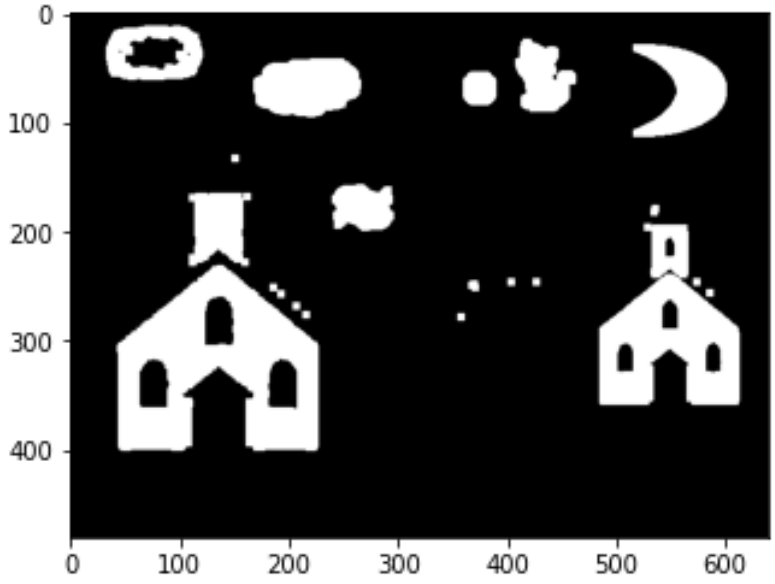
```
# Đặt ra threshold để làm nổi bật sự khác nhau
thresh = cv2.threshold(diff, 20, 255, cv2.THRESH_BINARY)[1]
plt.imshow(thresh, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7fbb3c4f60d0>
```



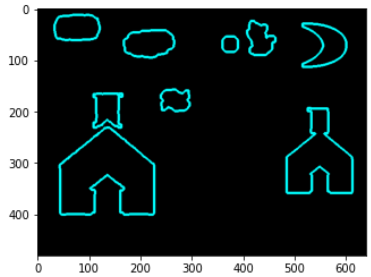

```
# Sử dụng dilate để nối các phần bị rời rạc
kernel = np.ones((3,3), np.uint8)
dilate = cv2.dilate(thresh, kernel, iterations=3)
plt.imshow(dilate, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7fbb3c3dd3d0>
```



```python
# Tìm contour của điểm khác biệt giữa 2 ảnh
mask = np.zeros(shape = image1.shape)

contours, _ = cv2.findContours(dilate, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
for contour in contours:
    area = cv2.contourArea(contour)

    # Nếu diện tích của sự khác biệt là nhỏ thì có thể là nhiễu nên ta sẽ bỏ qua
    if area > 100:
        cv2.drawContours(mask, [contour], -1, (0, 255, 255), 3)

        # Khoanh tròn các điểm khác biệt
        ((cx, cy), radius) = cv2.minEnclosingCircle(contour)
        cv2.circle(image1, (int(cx), int(cy)), int(radius), (0, 255, 255), 3)
        cv2.circle(image2, (int(cx), int(cy)), int(radius), (0, 255, 255), 3)

plt.imshow(mask, cmap = 'gray')
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
<matplotlib.image.AxesImage at 0x7fbb3c459430>
```




```python
plt.figure(figsize = (15, 15))
plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(image1, cv2.COLOR_BGR2RGB))
plt.subplot(1, 2, 2)
plt.imshow(cv2.cvtColor(image2, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7fbb3c9572b0>
```