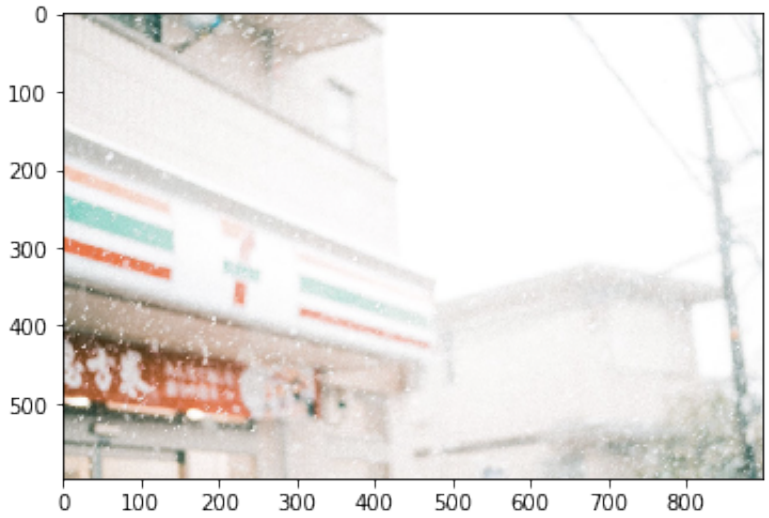
In [89]:
```
import numpy as np
import matplotlib.pyplot as plt
from skimage import draw
import cv2
import time
```

In [119]:
```
im = cv2.imread("3.png")
im = cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
plt.imshow(im)
im_average = im[:,:,0]/3 + im[:,:,1]/3 + im[:,:,2]/3
print(im.shape)
```

```
(596, 898, 3)
```



In [109]:
```
def calc_energy(im_average):
    dx_filter = np.array([1, -1])
    dy_filter = np.array([[1], [-1]])

    dx = cv2.filter2D(im_average, -1, dx_filter)
    dy = cv2.filter2D(im_average, -1, dy_filter)

    energy = np.zeros(dx.shape)
    energy = np.abs(dx)+np.abs(dy)
    return energy
```

In [110]:
```
def find_paths(im_average, index_map=None):
    energy = calc_energy(im_average)
    energy_map = np.zeros(energy.shape)
    (m, n) = energy_map.shape
    path = np.zeros((m-1,n, 2), dtype=int)
    energy_map[0,:] = energy[0,:]

    for i in range(1, m):
        min_i = np.argmin(np.array([energy_map[i-1][0], energy_map[i-1][1
        energy_map[i][0] = energy[i][0] + energy_map[i-1][min_i]

        min_i = np.argmin(np.array([energy_map[i-1][-1], energy_map[i-1][
        energy_map[i][-1] = energy[i][-1] + energy_map[i-1][-min_i-1]

        path[i-1][0][0] = min_i
        path[i-1][-1][0] = -min_i-1
        if index_map is not None:
            path[i-1][0][1] = index_map[i-1][min_i]
            path[i-1][-1][1] = index_map[i-1][-min_i-1]

        for j in range(1, n-1):
            min_i = np.argmin(np.array([energy_map[i-1][j-1], energy_map[
            energy_map[i][j] = energy[i][j] + energy_map[i-1][j-1+min_i]
            path[i-1][j][0] = j-1+min_i
            if index_map is not None:
                path[i-1][j][1] = index_map[i-1][j-1+min_i]

    return path, energy_map
```

Expanding

In [111]:
```
(m, n,_) = im.shape

paths = np.zeros((m, 100), dtype=np.intc)
i_map = np.repeat([np.arange(n)],[m], axis=0)

for i in range(50):

    path, energy_map = find_paths(im_average, index_map=i_map)

    min_energy = np.argmin(energy_map[-1,:])

    paths[-1][i] = min_energy
    prev = min_energy

    print(path.shape)
```

```
    for j in range(m-2, -1, -1):
        paths[j][i] = path[j][prev][1]
        prev = path[j][prev][0]

    mask = np.ones((m, n-i, 3), dtype = bool)

    for j in range(m-1, -1, -1):
        mask[j, paths[j][i], :] = np.zeros(3, dtype=bool)

    im_average = im_average[mask[:,:,0]].reshape((m, n-i-1))
    i_map = i_map[mask[:,:,0]].reshape((m, n-i-1))
    print(i)
```

```
(595, 898, 2)
0
(595, 897, 2)
1
(595, 896, 2)
2
(595, 895, 2)
3
(595, 894, 2)
4
(595, 893, 2)
5
(595, 892, 2)
6
(595, 891, 2)
7

(595, 890, 2)
8
(595, 889, 2)
9
(595, 888, 2)
10
(595, 887, 2)
11
(595, 886, 2)
12
(595, 885, 2)
13
(595, 884, 2)
14
(595, 883, 2)
15
(595, 882, 2)
16
(595, 881, 2)
17
```

```
(595, 880, 2)
18
(595, 879, 2)
19
(595, 878, 2)
20
(595, 877, 2)
21
(595, 876, 2)
22
(595, 875, 2)
23
(595, 874, 2)
24
(595, 873, 2)
25
(595, 872, 2)
26
(595, 871, 2)
27
(595, 870, 2)
28
(595, 869, 2)
29
(595, 868, 2)
30
(595, 867, 2)
31
(595, 866, 2)
32
(595, 865, 2)
33
(595, 864, 2)
34
(595, 863, 2)
35
(595, 862, 2)
36
(595, 861, 2)
37
(595, 860, 2)
38
(595, 859, 2)
39
(595, 858, 2)
40
(595, 857, 2)
41
(595, 856, 2)
42
```

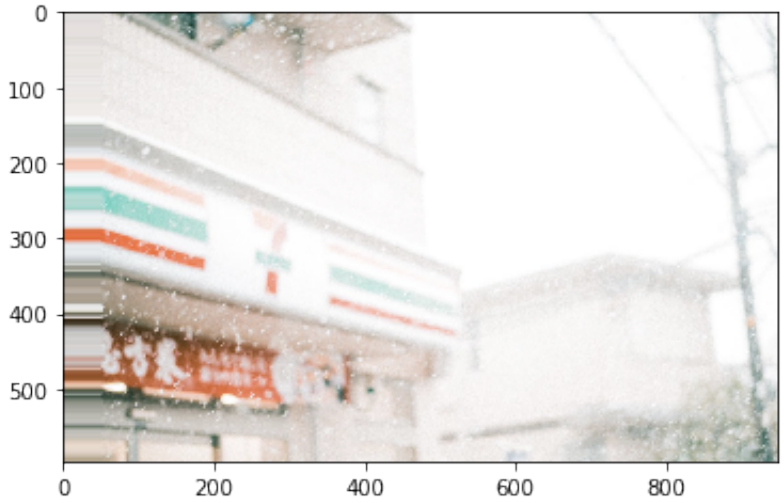
```
(595, 855, 2)
43
(595, 854, 2)
44
(595, 853, 2)
45
(595, 852, 2)
46
(595, 851, 2)
47
(595, 850, 2)
48
(595, 849, 2)
49
```

In [115]:
```
seam = np.zeros((m, n+50, 3), dtype = np.uint8)
mask = np.ones((m, n+50, 3), dtype = bool)

for i in range(m-1, -1, -1):
    line = paths[i, :]
    line.sort()
    for j in range(50):
        if line[j] == n-1:
            seam[i, int(line[j])+j, :] = im[i, int(line[j]), :]
        else:
            seam[i, int(line[j])+j, :] = (im[i, int(line[j]), :]/2 + im[i
        mask[i, int(line[j])+j, :] = np.zeros(3, dtype=bool)
```

In [116]:
```
out_img = np.zeros((m, n+50, 3), dtype=np.uint8)
nz = np.where(seam)
out_img[mask] = im.ravel()
out_img[nz] = seam[nz]
```

In [117]:
```
plt.imshow(out_img)
```

Out[117]: <matplotlib.image.AxesImage at 0xa21104f60>



Shrinking

In [*]:
```
out = im.copy()

for i in range(50):

    path, energy_map = find_paths(im_average)

    min_energy = np.argmin(energy_map[-1,:])

    paths = np.zeros(m, dtype=int)
    paths[-1] = min_energy


    for j in range(m-2, -1, -1):
        paths[j] = path[j][int(paths[j+1])][0]

    mask = np.ones((m, n-i, 3), dtype = bool)

    for j in range(m-1, -1, -1):
        line = paths[j]
        mask[j, line, :] = np.zeros(3, dtype=bool)

    out = out[mask].reshape((m, n-i-1, 3))
    im_average = im_average[mask[:,:,0]].reshape((m, n-i-1))
```

In [*]:
```
plt.imshow(out)
```

Keep Salient Object

In [21]:
```
def calc_energy_salient(im):
    (m, n, _) = im.shape
    mean = np.sum(im, axis=0)/m
    mean = np.sum(mean, axis=0)/n

    return np.linalg.norm(im-mean, ord=2, axis=2)
```

In [22]:
```
def find_paths_salient(im, energy, index_map=None):
    energy_map = np.zeros(energy.shape)
    (m, n) = energy_map.shape
    path = np.zeros((m-1,n), dtype=int)
    energy_map[0,:] = energy[0, :]

    #print(np.linalg.norm(im[100][101]-im[100][99]))

    for i in range (1, m):
        cu = np.linalg.norm(im[i][2]-im[i][0])
        cr = np.linalg.norm(im[i-1][1]-im[i][2]) + cu
        min_i = np.argmin(np.array([energy_map[i-1][1]+cu, energy_map[i-1
        energy_map[i][1] = energy[i][1] + energy_map[i-1][min_i+1]

        cu = np.linalg.norm(im[i][-1]-im[i][-3])
        cl = np.linalg.norm(im[i-1][-2]-im[i][-3]) + cu
        min_i = np.argmin(np.array([energy_map[i-1][-2]+cu, energy_map[i-
        energy_map[i][-2] = energy[i][-2] + energy_map[i-1][-min_i-2]

        if index_map is not None:
            path[i-1][1] = index_map[i-1][min_i+1]
            path[i-1][-2] = index_map[i-1][-min_i-2]
        else:
            path[i-1][1] = min_i+1
            path[i-1][-2] = -min_i-2
        for j in range(2, n-2):
            cu = np.linalg.norm(im[i][j+1]-im[i][j-1])
            cl = np.linalg.norm(im[i-1][j]-im[i][j-1]) + cu
            cr = np.linalg.norm(im[i-1][j]-im[i][j+1]) + cu
            min_i = np.argmin(np.array([energy_map[i-1][j-1]+cl, energy_m
            energy_map[i][j] = energy[i][j] + energy_map[i-1][j-1+min_i]
            if index_map is not None:
                path[i-1][j] = index_map[j-1+min_i]
            else:
                path[i-1][j] = j - 1 + min_i
    return path, energy_map
```

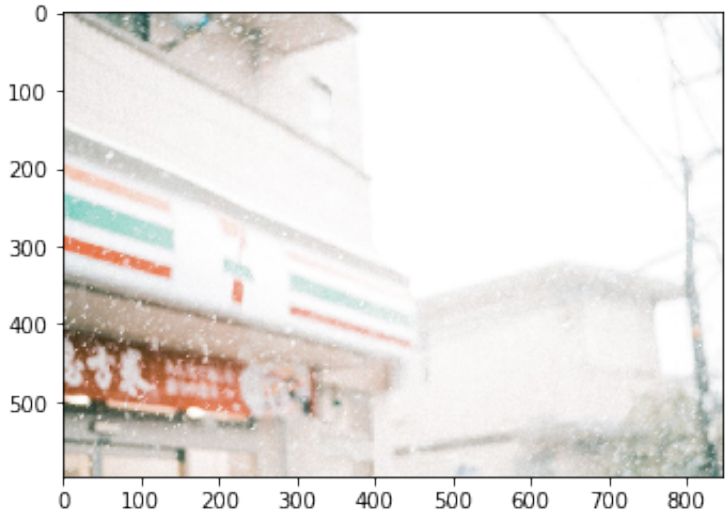
In [95]:
```
im = cv2.cvtColor(cv2.imread('3.png'), cv2.COLOR_BGR2RGB)
```

In [96]:
```
out = im.copy()
(m, n, _) = im.shape
energy = calc_energy_salient(im)
print(out.shape)

start = time.time()
for i in range(50):
    path, energy_map = find_paths_salient(out, energy)
    #print(energy_map)

    min_energy = np.argmin(energy_map[-1,1:-1])
    #print(min_energy)

    paths = np.zeros(m, dtype=int)
    paths[-1] = min_energy


    for j in range(m-2, -1, -1):
        paths[j] = path[j][int(paths[j+1])]
    #print(paths)

    mask = np.ones((m, n-i, 3), dtype = bool)

    for j in range(m-1, -1, -1):
        line = paths[j]
        mask[j, line, :] = np.zeros(3, dtype=bool)

    out = out[mask].reshape((m, n-i-1, 3))
    energy = energy[mask[:,:,0]].reshape((m, n-i-1))
    #print(i)

print("time took: ", time.time()-start)
```

```
(596, 898, 3)
time took:  693.6529970169067
```

In [97]:
```
plt.imshow(out)
```

Out[97]: <matplotlib.image.AxesImage at 0xa23f62278>



Faster Seam Carving

In [98]:
```
def find_min_window(energy_map, window_size):
    (h, w) = energy_map.shape
    min_energy = []
    for i in range(0, w-window_size+1):
        m = 0
        for j in range(window_size):
            m += sum(energy_map[:, i+j])
        min_energy.append(m)

    index = min_energy.index(min(min_energy))
    min_energy_window = energy_map[:, index:index+window_size]
    return min_energy_window
```

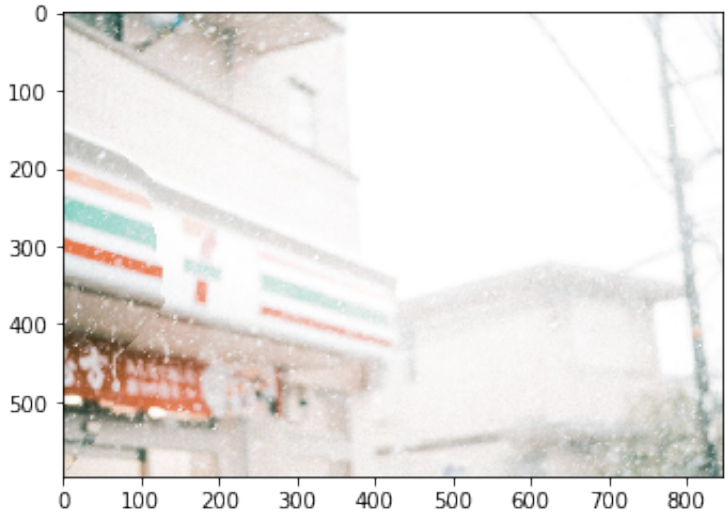
In [99]:
```
def remove_seams(im, energy, path, window, size):
    (m, n, _) = im.shape
    for i in range(size):
        min_energy = np.argmin(window[-1,1:-1])

        paths = np.zeros(m, dtype=int)
        paths[-1] = min_energy

        for j in range(m-2, -1, -1):
            paths[j] = path[j][int(paths[j+1])]

        mask = np.ones((m, n-i, 3), dtype = bool)

        for j in range(m-1, -1, -1):
            line = paths[j]
            mask[j, line, :] = np.zeros(3, dtype=bool)

        im = im[mask].reshape((m, n-i-1, 3))
        energy = energy[mask[:,:,0]].reshape((m, n-i-1))

    return im
```

In [104]:
```
im = cv2.cvtColor(cv2.imread('3.png'), cv2.COLOR_BGR2RGB)
out = im.copy()

remove = 50
window_size = 20
half_window = window_size // 2
iteration = remove // half_window
remain = remove % half_window

start = time.time()
if(iteration > 0):
    for i in range(iteration):
        energy = calc_energy_salient(out)
        path, energy_map = find_paths_salient(out, energy)
        min_energy_window = find_min_window(energy_map, window_size)
        out = remove_seams(out, energy, path, min_energy_window, half_win

print("time took: ", time.time()-start)
```

```
time took:  76.60574102401733
```

In [105]:
```
plt.imshow(out)
```

Out[105]: <matplotlib.image.AxesImage at 0xa24b81978>



Object Removal

In [72]:
```
def poly2mask(vertex_row_coords, vertex_col_coords, shape):
    fill_row_coords, fill_col_coords = draw.polygon(vertex_row_coords, ve
    mask = np.zeros(shape, dtype=np.bool)
    mask[fill_row_coords, fill_col_coords] = True
    return mask
```

In [73]:
```
def specify_mask(img):
    # get mask
    print("If it doesn't get you to the drawing mode, then rerun this fun
    fig = plt.figure()
    fig.set_label('Draw polygon around source object')
    plt.axis('off')
    plt.imshow(img, cmap='gray')
    xs = []
    ys = []
    clicked = []

    def on_mouse_pressed(event):
        x = event.xdata
        y = event.ydata
        xs.append(x)
        ys.append(y)
        plt.plot(x, y, 'r+')

    def onclose(event):
        clicked.append(xs)
        clicked.append(ys)
    # Create an hard reference to the callback not to be cleared by the g
    # collector
    fig.canvas.mpl_connect('button_press_event', on_mouse_pressed)
    fig.canvas.mpl_connect('close_event', onclose)
    return clicked
```

In [74]:
```
def get_mask(ys, xs, img):
    mask = poly2mask(ys, xs, img.shape[:2]).astype(int)
    fig = plt.figure()
    plt.imshow(mask, cmap='gray')
    return mask
```

In [75]:
```
def get_gray_img(img):
    d = img.ndim
    if(d == 3):
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    else:
        gray = img

    return gray
```

Choose Removal Object

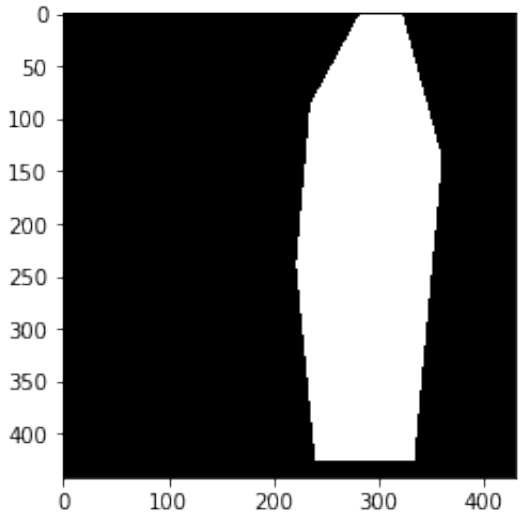
In [76]:
```
object_img = cv2.cvtColor(cv2.imread('10.png'), cv2.COLOR_BGR2RGB)
object_gray = get_gray_img(object_img)
```

In [80]:
```
%matplotlib notebook
mask_coords = specify_mask(object_img)
```

```
If it doesn't get you to the drawing mode, then rerun this function ag
ain.
```

In [81]:
```
xs = mask_coords[0]
ys = mask_coords[1]
%matplotlib inline
plt.figure()
mask_remove = get_mask(ys, xs, object_img)
```

```
<matplotlib.figure.Figure at 0x1093ac5f8>
```



In [82]:
```
_, energy_map = find_paths(object_gray)
```

In [83]:
```
out = object_img.copy()
(m, n, _) = out.shape
energy = calc_energy_salient(object_img)

ys, xs = np.where(mask_remove == 1)
for i in range(len(xs)):
    energy[int(ys[i]), int(xs[i])] = -255
```

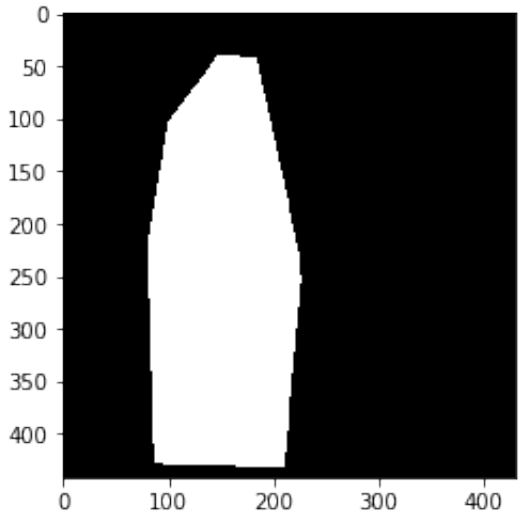
Choose Keep Object

In [84]:
```
%matplotlib notebook
mask_coords = specify_mask(object_img)
```

```
If it doesn't get you to the drawing mode, then rerun this function ag
ain.
```

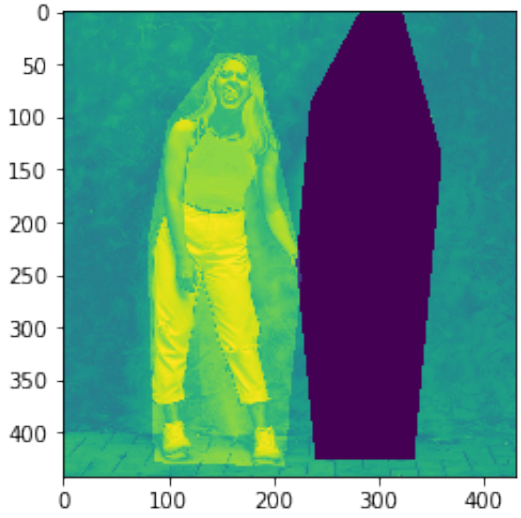
In [85]:
```
xs = mask_coords[0]
ys = mask_coords[1]
%matplotlib inline
plt.figure()
mask_keep = get_mask(ys, xs, object_img)
```

```
<matplotlib.figure.Figure at 0xa2552bb70>
```



In [86]:
```
ys, xs = np.where(mask_keep == 1)
for i in range(len(xs)):
    energy[int(ys[i]), int(xs[i])] += 100
plt.imshow(energy)
```

Out[86]:
```
<matplotlib.image.AxesImage at 0xa24c30080>
```

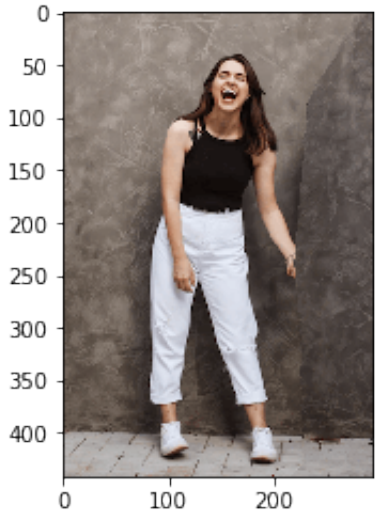


In [87]:
```
for i in range(135):
    path, energy_map = find_paths_salient(out, energy)
    #print(energy_map)

    min_energy = np.argmin(energy_map[-1,1:-1])
    #print(energy_map[-1,1:-1],min_energy)

    paths = np.zeros(m, dtype=int)
    paths[-1] = min_energy


    for j in range(m-2, -1, -1):
        paths[j] = path[j][int(paths[j+1])]
    #print(paths)

    mask = np.ones((m, n-i, 3), dtype = bool)

    for j in range(m-1, -1, -1):
        line = paths[j]
        mask[j, line, :] = np.zeros(3, dtype=bool)

    out = out[mask].reshape((m, n-i-1, 3))
    energy = energy[mask[:,:,0]].reshape((m, n-i-1))
    print(i)
```

```
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
```

In [88]:
```
plt.imshow(out)
```

Out[88]: <matplotlib.image.AxesImage at 0xa24ba8860>



In [ ]:
```

```