Image Analogy

In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np
import imageio
from main import *
```

Part 1. Basic Usage

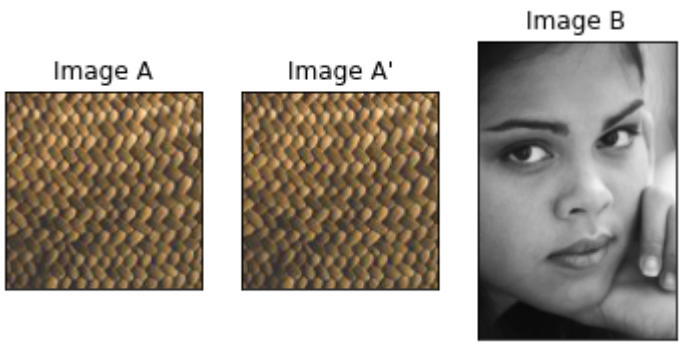
Texture Transfer

In [ ]:

```
a1_img_fn = 'images/transfer2_A1.jpg'
a1_img = np.float32(cv2.cvtColor(cv2.imread(a1_img_fn), cv2.COLOR_BGR2RGB)/255.0
)

a2_img_fn = 'images/transfer2_A2.jpg'
a2_img = np.float32(cv2.cvtColor(cv2.imread(a2_img_fn), cv2.COLOR_BGR2RGB)/255.0
)

b1_img_fn = 'images/transfer2_B1.jpg'
b1_img = np.float32(cv2.cvtColor(cv2.imread(b1_img_fn), cv2.COLOR_BGR2RGB)/255.0
)

fig, axes = plt.subplots(1, 3)
axes[0].imshow(a1_img)
axes[0].set_title('Image A'), axes[0].set_xticks([]), axes[0].set_yticks([])
axes[1].imshow(a2_img)
axes[1].set_title("Image A'"), axes[1].set_xticks([]), axes[1].set_yticks([])
axes[2].imshow(b1_img)
axes[2].set_title("Image B"), axes[2].set_xticks([]), axes[2].set_yticks([])
```

Out[ ]:

```
(Text(0.5, 1.0, 'Image B'), [], [])
```



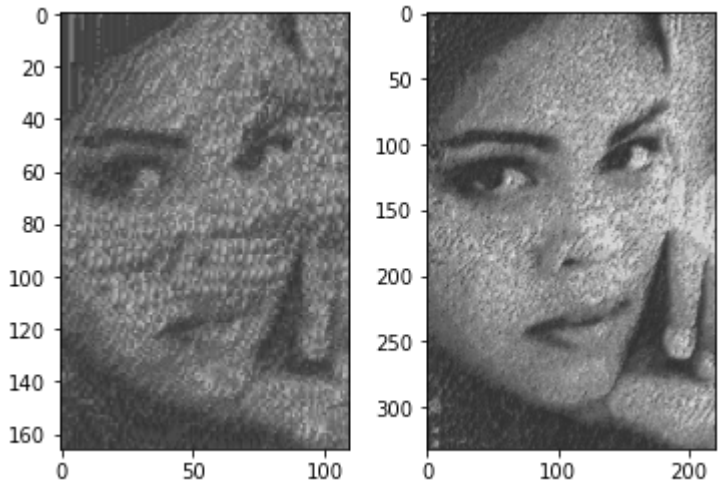
In [ ]:

```
kappa = 0.5
b2_img_list_trans = start(a1_img,b1_img,a2_img,kappa,False)
```

In [ ]:

```
length = len(b2_img_list_trans)

fig, axes = plt.subplots(1, length)
for i in range(length):
    axes[i].imshow(b2_img_list_trans[i])

imageio.imsave('output/texture_transfer.jpg', b2_img_list_trans[-1])
```

```
Lossy conversion from float32 to uint8. Range [0, 1]. Convert image
to uint8 prior to saving to suppress this warning.
```



Super-resolution

In [ ]:

```
a1_img_fn = 'images/spResA1.jpg'
a1_img = np.float32(cv2.cvtColor(cv2.imread(a1_img_fn), cv2.COLOR_BGR2RGB)/255.0
)

a2_img_fn = 'images/spResA2.jpg'
a2_img = np.float32(cv2.cvtColor(cv2.imread(a2_img_fn), cv2.COLOR_BGR2RGB)/255.0
)

b1_img_fn = 'images/spResB1.jpg'
b1_img = np.float32(cv2.cvtColor(cv2.imread(b1_img_fn), cv2.COLOR_BGR2RGB)/255.0
)

fig, axes = plt.subplots(1, 3)
axes[0].imshow(a1_img)
axes[0].set_title('Image A'), axes[0].set_xticks([]), axes[0].set_yticks([])
axes[1].imshow(a2_img)
axes[1].set_title("Image A'"), axes[1].set_xticks([]), axes[1].set_yticks([])
axes[2].imshow(b1_img)
axes[2].set_title("Image B"), axes[2].set_xticks([]), axes[2].set_yticks([])
```

Out[ ]:

```
(Text(0.5, 1.0, 'Image B'), [], [])
```

In [ ]:

```
a1_img=resize_img(a1_img,1)
a2_img=resize_img(a2_img,1)
b1_img=resize_img(b1_img,1)
```

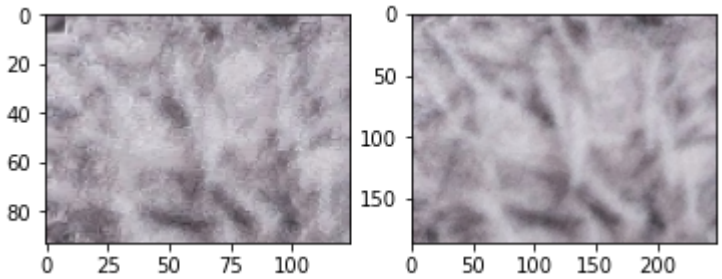
In [ ]:

```
kappa = 2
b2_img_list_sup = start(a1_img,b1_img,a2_img,kappa,False)
```

In [ ]:

```
length = len(b2_img_list_sup)

fig, axes = plt.subplots(1, length)
for i in range(length):
    axes[i].imshow(b2_img_list_sup[i])

imageio.imsave('output/supRes.jpg', b2_img_list_sup[-1])
```

```
Lossy conversion from float32 to uint8. Range [0, 1]. Convert image
to uint8 prior to saving to suppress this warning.
```



Part 3. Texture-by-numbers

Example from paper

In [ ]:

```
a1_img_fn = 'images/drawA1.jpg'
a1_img = np.float32(cv2.cvtColor(cv2.imread(a1_img_fn), cv2.COLOR_BGR2RGB)/255.0
)

a2_img_fn = 'images/drawA2.jpg'
a2_img = np.float32(cv2.cvtColor(cv2.imread(a2_img_fn), cv2.COLOR_BGR2RGB)/255.0
)

b1_img_fn = 'images/drawB1.jpg'
b1_img = np.float32(cv2.cvtColor(cv2.imread(b1_img_fn), cv2.COLOR_BGR2RGB)/255.0
)

a1_img=resize_img(a1_img,1)
a2_img=resize_img(a2_img,1)
b1_img=resize_img(b1_img,1)

fig, axes = plt.subplots(1, 3)
axes[0].imshow(a1_img)
axes[0].set_title('Image A'), axes[0].set_xticks([]), axes[0].set_yticks([])
axes[1].imshow(a2_img)
axes[1].set_title("Image A'"), axes[1].set_xticks([]), axes[1].set_yticks([])
axes[2].imshow(b1_img)
axes[2].set_title("Image B"), axes[2].set_xticks([]), axes[2].set_yticks([])
```

Out[ ]:

```
(Text(0.5, 1.0, 'Image B'), [], [])
```

In [ ]:

```
kappa = 1.5
b2_img_list_hddraw = start2(a1_img,b1_img,a2_img,kappa,True)
```

In [ ]:

```
length = len(b2_img_list_hddraw)

fig, axes = plt.subplots(1, length)
for i in range(length):
    axes[i].imshow(b2_img_list_hddraw[i])

imageio.imsave('output/hddraw.jpg', b2_img_list_hddraw[-1])
```

```
Lossy conversion from float32 to uint8. Range [0, 1]. Convert image
to uint8 prior to saving to suppress this warning.
```